SUBSIDISED LOW COST EDITIONS

LCSE

NBT. INDIA

912

கணிப்பொறி இயலுக்கு அறிமுகம்

முனைவர் வெ. கிருஷ்ணமூர்த்தி

அண்ணா பல்கலைக் கழகம்

சென்னை - 600 025.

பொறியியல் பாடநூல் 3

கணிப்பொறி இயலுக்கு அறிமுகம்

முனைவர் வெ. கிருஷ்ணமூர்த்தி
பேராசிரியர்

கணிப்பொறி அறிவியல்
தொழில் நுட்பத்துறை
அண்ணா பல்கலைக்கழகம்

அண்ணா பல்கலைக்கழகம்
சென்னை – 600 025

முதற் பதிப்பு

திருவள்ளுவராண்டு 2028, மாசி மாதம் மார்ச், 1997
பதிப்பு வகை : மாணவர் சலுகைப் பதிப்பு
பதிப்புரிமை : பதிவாளர்
அண்ணா பல்கலைக்கழகம்
சென்னை 600 025

விலை ரூ. 70.00

ISBN: 81 - 85845 - 07 - 7

This book has been subsided by the Government of India through the National Book Trust, India for the benefit of students.

தேசிய புத்தக நிறுவன நிதிச்சலுகை பெற்ற வெளியீடு

விற்பனை உரிமை : பதிவாளர்
அண்ணா பல்கலைக் கழகம்
சென்னை 600 025

பணி ஒருங்கிணைப்பு : வளர் தமிழ் மன்றம்
அண்ணா பல்கலைக் கழகம்
சென்னை 600 025

படங்கள் : ஷியாம்

அச்சிட்டோர் பாவை அச்சகம்
142, சானி சான் கான் சாலை
இராயப்பேட்டை, சென்னை 600 014.

அணிந்துரை

கணிப்பொறிக் கல்வி என்பது இன்று எல்லா நிலைகளிலும் ஓர் அடிப்படைத் தேவையாக உருவாகி வருகிறது. குறிப்பாகப் பொறியியல் துறையில் இன்றியமையாததோர் இடத்தைப் பெற்றுள்ளது. எனவே அண்ணா பல்கலைக்கழகத்தின் வளர்தமிழ் மன்ற வழித் தமிழ்ப் பாடநூல் வெளியீட்டுத் திட்டத்தின் முதல் கட்ட நூல்களில் ஒன்றாக இந்நூல் எடுத்துக்கொள்ளப்பட்டது.

கணிப்பொறிக் கல்வியில் பல்லாண்டுகளாகப் பணிபுரிந்து வரும் பேராசிரியர் முனைவர் வெ. கிருட்டிணமூர்த்தி அவர்கள் இந்நூலை முதலாண்டு பட்டயக் கல்வி மாணவர்களின் தேவையையொட்டி எழுதியுள்ளார். எளிய நடையில் கணிப்பொறியின் அடிப்படைகள், பொறியமைப்பு, கணிப்பு, வரைவியல்,பயன்பாடுகள் ஆகியவற்றைத் தெளிவாக விளக்கியுள்ளார்.

ஆங்கிலம் அறியாமல் தமிழில் மட்டுமே தேர்ச்சி பெற்ற கோடிக்கணக்கானவர் தமிழ்நாட்டில் இருக்கின்றார்கள். வளர்ந்துவரும் கணிப்பொறியின் தொழில் நுட்பங்களான இண்டர்நெட், கணிப்பொறிவழிக் கல்வி போன்றவற்றின் பயன்களை இவர்களும் அடைய முடிந்தால்தான் தமிழகம் முன்னேற முடியும். இதற்குத் தமிழ் மட்டுமே அறிந்தவர்களும் கணிப்பொறி பற்றித் தெரிந்துகொள்வதும், கணிப்பொறியுடன் தமிழிலேயே உரையாடி அதன் பயன்களைப் பெறுவதும் வேண்டும். இந்த நோக்கில், தமிழுக்கென ஓர் எளிமையான சாவிப்பலகை உருவாக்கம், கலைச் சொல்லாக்கம், தமிழ் கற்பிக்கும் மென்பொருட்களின் வடிவமைப்பு போன்ற பலவகைகளிலும் சிறப்பாகப் பணியாற்றுவதுடன், கணிப்பொறிக்கான இந்த இனிய அறிமுக நூலையும் எழுதி யுள்ளார் இதன் ஆசிரியர். இந்நூல் கணிப்பொறி பற்றிக் கற்க விழையும் அனைவருக்கும் ஒரு தொடக்க நூலாக விளங்கம் என்பதில் ஐயமில்லை.

பொறியியல் உயர் கல்வியைத் தமிழ் வழி அளிக்கும் சீரிய முயற்சிகளுக்கு இந்நூல் உறுதுணையாக விளங்கும். தமிழக மக்கள், குறிப்பாகப் பொறியியல் மாணவர்களும் ஆசிரியர்களும், இந்நூலைப் பெரிதும் வரவேற்பார்கள் என நம்புகிறேன். இந்நூல் அனைவர்க்கும் மலிவு விலையில் கிடைக்க வேண்டிப் பொருளுதவி செய்யும் தேசியப் புத்தக நிறுவனத்தின் சேவையும் பாராட்டி நன்றி கூர்தற்குரியது.

சென்னை - 25
28 - 2 - 1997

இரா.மா. வாசகம்
துணைவேந்தர்
அண்ணா பல்கலைக்கழகம்

பதிப்புரை

மணக்க வரும் தென்றலிலே குளிரா இல்லை
மாந்தோப்பில் நிழலா இல்லை
தணிப்பரிதாம் துன்பமிது
தமிழ்நாட்டின் தனித்தெருவில் தமிழ்தான் இல்லை
என்று பாடினார் பாவேந்தர் பாரதிதாசன்.

இந்நிலை களையப்பட வேண்டும் என்பதிலும், அதற்குத் துணை புரிய தாய்மொழிவழிக் கல்வியே தலைசிறந்தது என்பதிலும் கருத்து வேறுபாடுகளுக்கு இடமில்லை. இந்த அடிப்படையில் தமிழக அரசு தமிழ்வழியில் பொறியியல், மருத்துவப் பாடங்களை நடத்தவேண்டும் என்ற திட்டத்தை அறிவித்து அதற்காக அண்ணா பல்கலைக் கழகத்திற்கு நிதி ஒதுக்கீடு செய்து பொறியியல் பாடநூல்களை எழுதச் செய்து வெளியிடுமாறு பணித்தது. அதன்படி முனைவர் அ. இளங்கோவன் அவர்கள் எழுதிய "திண்ம விசையியல்" நூலும், முனைவர் மு. ஆறுமுகம் எழுதிய "பொருள்களின் அறிவியல்" நூலும் அண்ணா பல்கலைக்கழகத்தால் வெளியிடப்பட்டுள்ளன. மூன்றாவது நூலாக முனைவர் வெ. கிருட்டிணமூர்த்தியின் "கணிப்பொறி இயலுக்கு அறிமுகம்" இப்போது வெளியிடப்படுகிறது. இனியும் பல நூல்கள் வெளி வர உள்ளன.

இந்த நூலை எழுதிய நூலாசிரியருக்கு முதற்கண் என் பாராட்டுக்கள். இந்த நூலை வெளிக்கொணர்வதற்கான பணிகளை ஒருங்கிணைத்தது அண்ணா பல்கலைக்கழக வளர் தமிழ் மன்றத்தினர். வெளியீட்டுப் பணிகளை செவ்வனே நடத்திக் கொடுத்தவர்கள், இதற்கென அமைக்கப்பட்ட வழி நடத்தும் குழுவினர்.

தமிழை அறிவியல் பாடமொழி ஆக்க வேண்டும் என்பதில் தணியாத பற்றுக் கொண்டவர் அண்ணா பல்கலைக்கழகத் துணைவேந்தர் பேராசிரியர் இரா.மா. வாசகம் அவர்கள். அவர்களது ஆர்வமும் வழிகாட்டுதலும் இந்த நூல் செம்மையாக அமையப் பெரிதும் உதவின. அத்தோடு மட்டுமின்றி அருமையான அணிந்துரை ஒன்றையும் வழங்கியுள்ளார்.

இந்த நூலுக்கு மூல நிதி உதவி அளித்தது தமிழக அரசு. இந்த நூலை மாணவர் பதிப்பாக மலிவு விலையில் வெளியிட பல்கலைக்கழகத்திற்கு நூல் மதிப்பீட்டில் 50 விழுக்காடும், நூலாசிரியருக்கு 20 விழுக்காடு மதிப்பூதியமும் தர ஒப்புக் கொண்டுள்ளது தேசிய புத்தக நிறுவனம்.

இவ்வாறு பலருடைய கூட்டு முயற்சியினால் உருவாகியுள்ளது இந்நூல். இதில் பங்கேற்ற அனைவரும் பாராட்டுக்கும், நன்றிக்கும் உரியவர்கள்.

தமிழ்வழிக் கல்விக்குத் தடம் அமைப்பதில் அண்ணா பல்கலைக்கழகம் ஆற்றவரும் இத்தகைய சீரிய பணிகளுக்குத் தமிழக மக்களும், கல்வியாளர்களும் உறுதுணையாக இருப்பார்கள் என்று நம்புகின்றேன்.

சென்னை - 25
3 - 3 - 1997

முனைவர் இரா. செகதீசன்
பதிவாளர்
அண்ணா பல்கலைக்கழகம்

முகவுரை

இப்புத்தகம் பல்தொழிற் பயிற்சி நிலையங்களில் கணிப்பொறித் துறைக்கான ஒரு பாட நூலாக எழுதப்பட்டுள்ளது. கணிப்பொறி பற்றிய ஓர் அறிமுகம் இப்புத்தகத்தின் முதற்பாகமாக அமைந்துள்ளது. மூன்று பகுதிகளைக் கொண்ட இப்பாகம் கணிப்பொறியின் வரலாறு, அதன் பாகங்கள், அதில் பயன்படும் அடிப்படைக் கணிதம், அது செயல்படும் விதம், மொழிகளின் வளர்ச்சி, தீர்வு காணும் முறைகள் எனப் பல கூறுகளைக் கொண்டது.

இரண்டாம் பாகத்தில் பேசிக் மொழி அறிமுகப்படுத்தப்படுகிறது. இது படிப்படியாக முன்னேறும் வகையில், நான்கு பகுதிகளாக எழுதப்பட்டுள்ளது; அடிப்படைக் கூறுகள், துணைமுறைகள், கோப்புகள், வரைவியலுக்கு ஓர் அறிமுகம் எனப் பிரித்து எழுதப்பட்டுள்ளது.

இரு நோக்கங்களுடன் இப்புத்தகம் எழுதப்பட்டுள்ளது. படிப்பதற்கு எளிமையாகவும் இனிமையாகவும் இருக்க வேண்டும் என்பது முதல் நோக்கம். இதற்காக எளிய தமிழ்ச் சொற்கள், சில சமயங்களில் பேச்சு வழக்கில் உள்ளவை. பயன்படுத்தப்பட்டுள்ளன. மேலும் கருத்துக்களைத் தெளிவுபடுத்தவும் மனத்தில் எளிதில் பதியவும் பல 'கார்ட்டூன்' படங்கள் இடம் பெற்றுள்ளன.

இப்புத்தகத்தில் பேசிக் மொழியின் அறிமுகம் ஒரு பெரும் பகுதியை வகிக்கிறது. கணிப்பொறிகள் வரத் தொடங்கிய காலத்தில் தோன்றிய முதல் மொழிகளில் ஒன்று பேசிக். இன்று (1997) பொருள் சார் வடிவமைப்பு வழியில் செயல்படும் சி ++, ஜாவா போன்ற மொழிகள் பெரிதும் பயன்படுகின்றன. எனவே பேசிக் மொழியைப் படிப்பதை ஒரு முடிவாகக் கருதாமல், ஒரு தொடக்கமாகக் கருத வேண்டும். அந்த நோக்கத்தில் எந்த மொழியையும் சாராமல் கணக்குகளைத் தீர்க்கும் முறைகளை அறிவது அவசியம். இதற்காக ஒரு சிறந்த முறையாக போலிக் குறியீட்டு முறை அறிமுகப்படுத்தப்பட்டுள்ளது. மாணவர்கள் பல மொழிகளிலும் நிரல்கள் எழுதுவதை எளிதாக்கும் முறைகளை அறிந்து கொள்ளச் செய்வது இங்கு இரண்டாவது நோக்கம்.

இந்தப் புத்தகம் வெளிவரப் பலரும் பல விதங்களில் துணை புரிந்துள்ளனர். குறைகளை எடுத்துக் காட்டியும், படிகளைத் திருத்தியும் உதவியது என் மனைவி திருமதி கமலா. தமிழை மேம்படுத்த பல ஆலோசனைகளைக் கூறியவர் பேராசிரியர் இராம. சுப்பிரமணியன். என் கருத்துகளைக் கேட்டு அவற்றுக்கு உயிரூட்டும் வகையில் அழகான படங்களை வரைந்து கொடுத்தவர் திரு ஷியாம். தேவையான பணிகளை ஒருங்கிணைத்த சிறப்பு அலுவலர்கள் திரு. கண்ணனும், திரு. இராமலிங்கமும், முனைவர் தமிழப்பனும். இப்புத்தகத்தினைச் சிறந்த முறையில் அச்சிட்டுக் கொடுத்தவர்கள் பாவை அச்சகத்தினர்.

பேராசிரியர்கள் அ. இளங்கோவன், நா. சங்கரநாராயணன் மற்றும் திரு. மணவை முஸ்தபா போன்றோர் பலவிதங்களிலும் துணைபுரிந்துள்ளார்கள். மற்றும் மாணவர்களுக்குக் குறைந்த விலையில் இப்புத்தகம் கிடைக்க வேண்டும் என்று நிதிச் சலுகை அளித்தவர்கள் தேசிய புத்தக நிறுவனத்தினர். இவர்களுக்கும் எனக்கு ஊக்கமளித்த நண்பர்கள் அனைவருக்கும் என் உளமார்ந்த நன்றி.

சென்னை - 25

3 -3 - 1997

வெ. கிருஷ்ணமூர்த்தி

பொருளடக்கம்

கணிப்பொறி வரலாறும் கணிப்பொறிக் கணிதமும்

1

கணிப்பொறியின் வரலாற்றையும், கணிப்பொறியில் பயன்படும் கணிதத்தின் அடிப்படையையும் தெரிந்துகொள்வது இந்தப் பகுதியின் நோக்கமாகும்.

ஆதிமனிதன் கணக்கிடத் தொடங்கிய காலம் முதல், மணிச்சட்டம், பெருக்கல் அட்டைகள், பல்சக்கரப் பொறிகள் என்று, கணிக்க உதவும் பொறிகள் வளர்ந்த வரலாற்றை முதலில் காண்போம். பிறகு, மின்சாரத்தால் இயங்கி, நகரும் பாகங்களைக் கொண்ட பொறிகளும், மின்னணு இயலைப் பயன்படுத்தி நகரும் பாகங்கள் எவையும் இல்லாமல் உருவாக்கப்பட்ட பொறிகளும் தோன்றிய வரலாற்றைப் பார்ப்போம். அடுத்து கணிப்பொறியின் தலைமுறையினையும், அதன் வகைகளையும் காண்போம். தொடர்ந்து கணிப்பொறியில் பயன்படும் கணிதத்தைச் சிறிது காண்போம். இருநிலை, எண்ணிலை மற்றும் பதினாறுநிலை எண்களையும், இருநிலை எண்களைக் கொண்டு கணிக்கும் முறையையும் பார்ப்போம். இம்முறையில் உள்ள சிக்கல்களைத் தீர்க்க உதவும் நிரப்புமுறைக் கணிதத்தையும், பின்ன எண்களைக் குறிப்பிட உதவும் மிதக்கும் புள்ளி முறையையும் கற்போம். எழுத்துக்களைக் குறிப்பிட உதவும் குறியீட்டு முறைகளையும், பூலியன் கணித அமைப்பையும் தெரிந்து கொள்வோம்.

1.1 கணிப்பொறியின் ஆதி வரலாறு

மனிதன் பொருள்களை ஒன்று, இரண்டு என்று எண்ணத் தொடங்கிய காலத்தையே கணிப்பொறி வரலாற்றின் தொடக்க காலமாகக் கருத வேண்டும்.

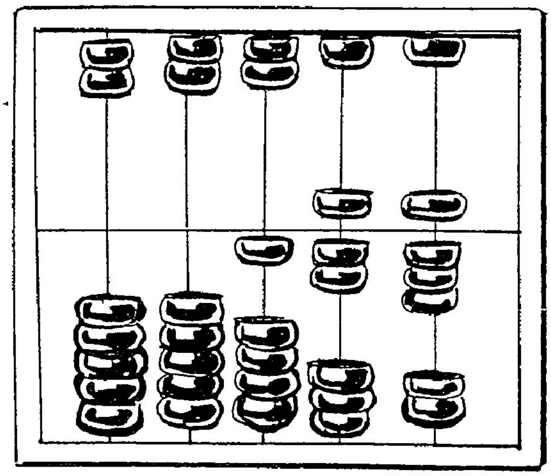
அன்றைய மனிதன் எண்ணுவதற்குச் சிறுசிறு கற்கள் போன்றவற்றைப் பயன்படுத்தினான். எண்களின் பயன்பாடு அதிகரிக்க அதிகரிக்க, பெரிய எண்களைக் கையாள வேண்டிய அவசியம் ஏற்பட்டது. எண்களைக் குறிக்கக் குறியீடுகள் தோன்றின; நம் கைகளில் பத்து விரல்கள் இருப்பதனால் தசமக் குறியீடுகள் தோன்றின. பாபிலோனியர் அறுபதை அடிப்படையாகக் கொண்ட எண்களைப் பயன்படுத்தியிருக்கின்றனர். இந்தியாவில் பதினாறின் அடிப்படையில் நாணயங்கள் புழங்கின (16 அணா = 1 ரூபாய்). நாளாகஆக, செய்யுவேண்டிய கணிப்புகளின் கடினத்தன்மை அதிகமாகிக் கொண்டே வந்தது. சில தகவல்கள் பல இடங்களிலும் பயன்பட்டமையால் அவற்றை விரைவாகவும் எளிதாகவும் கணிக்க பல யுக்திகள் கண்டுபிடிக்கப்பட்டன. இந்தியக் கணித மேதைகளான ஆர்யபட்டா, பாஸ்கரா போன்றவர்கள் எழுதி வைத்துள்ள வழிமுறைகள், இன்றைய கணிப்பொறிகளில் சிறப்பாகப் பயன்படும் என்று தோன்றுகிறது. அதனால் அவற்றைப் பற்றிப் பல்கலைக் கழகங்கள் இன்று ஆராய்ந்து வருகின்றன.

சுமேரிய நாகரிக காலத்தில் கணிக்கப்பட்ட விடைகள், பிற்காலத்தில் பயன்படுத்துவதற்காக, பட்டியல்களாக எழுதிவைக்கப்பட்டன. இவற்றில் சில களிமண் பலகைகளில் உள்ளன. காகிதம் கண்டுபிடிக்கப்பட்டவுடன் இம்மாதிரிப் பட்டியல்கள் அதில் எழுதிவைக்கப்பட்டன.

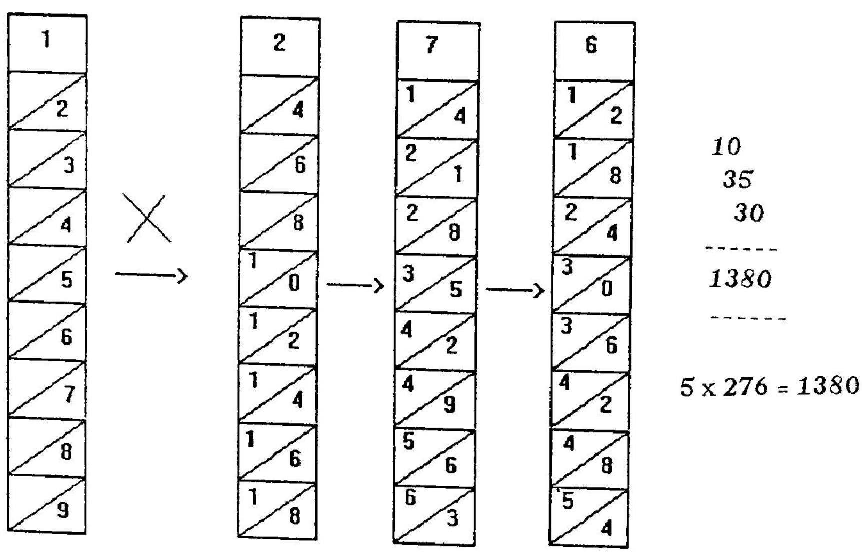
மணிச்சட்டம்

விரைவில் விடைகாணப் பயன்படுத்தப்பட்ட ஆதிகாலக் கருவிகளில் முக்கியமானது **அபாகஸ்** (Abacus) எனப்படும் **மணிச்சட்டம்** (படம் 1-1). இது குழந்தைகளுக்கான ஒரு கல்வி விளையாட்டுப் பொருளாக இப்போது கிடைக்கிறது. இதில் பல வகைகளுண்டு. படம் 1-1 இல் ஒரு வகை உள்ளது. இதில் எத்தனைச் சரங்கள் உள்ளன, ஒவ்வொரு சரத்திலும் எத்தனை மணிகள் உள்ளன, சட்டம் ஒரே பாகமாக உள்ளதா அல்லது இரு பாகங்களாக உள்ளதா, என்பன வற்றைப் பொறுத்து மணிச்சட்டம் பல வகைப்படும். படத்தில் உள்ள மணிச்சட்டத்தின் மேல்பாகத்தில் உள்ள ஒரு மணிக்கு மதிப்பு 5 ; கீழே உள்ள மணிக்கு மதிப்பு 1. ஒவ்வொரு சரமும்

அதன் இடத்தைப் பொறுத்து 1, 10, 100 என மதிப்புப் பெறும். படத்தில் 178 என்ற எண் காட்டப்பட்டிருக்கிறது. இன்றும் ஜப்பானில் மணிச்சட்டங்கள் பயன்படுத்தப் படுகின்றன. பயிற்சி பெற்றவர்கள் இவற்றை மிக விரைவாகப் பயன்படுத்து கிறார்கள்; கணிப்பான்களின் (Calculator) உதவியால் கணிப்பதைவிட விரைவில் கணித்து முடித்துவிடுகின்றனர்!



1.1 மணிச்சட்டம்



படம் 1.2 நேப்பியர் குச்சிகள்

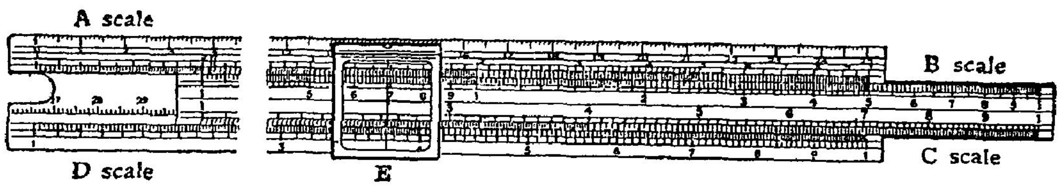
பெருக்கல் குச்சிகள்

அடுத்து வருவது லாகரிதம் கண்டுபிடித்த நேப்பியர் (John Napier) என்னும் ஸ்காட்லாந்துக்காரர் கண்டுபிடித்த **நேப்பியர் குச்சிகளாகும்** (Napier Bones). இது பெருக்கலுக்குப் பயன்படுத்தப்பட்ட ஒன்பது குச்சிகள் (அட்டைகள்) அடங்கிய தொகுதி. 276 x 5 என்ற பெருக்கலைச் செய்யும் விதத்தைப் படம் 1.2 காண்பிக் கிறது.

இன்றும்கூட மாணவர்கள் பெருக்கல் வகுத்தல்களை எளிதாகச் செய்ய நேப்பியர் லாகரிதப் பட்டியலைப் பயன்படுத்துகின்றனர். கணிப்பான் வந்த பிறகு இதன் மதிப்பு குறைந்துவிட்டது.

நகரும் சட்டம் (Slide rule)

17ஆம் நூற்றாண்டில் கண்டுபிடிக்கப்பட்ட நகரும் சட்டம் நேப்பியரின் லாகரிதத்தின் அடிப்படையில் அமைந்தது. கணிப்பான்கள் வரும்வரை எல்லாப் பொறியாளர்களின் கையிலும் காணப்பட்ட சிறுபொறி இது.



படம் 1-3 நகரும் சட்டம்

பல்சக்கரவகைக் கணிப்பான்கள்

பாஸ்கல், லைப்னீஸ், பாபேஜ் போன்றவர்கள் பல் சக்கரங்களால் இயங்கும் கணிப்பான்களை உருவாக்கினர். இவை கணிப்பொறிக்கு முன்னோடியான கால்குலேட்டர் (Calculator) வகையைச் சேர்ந்தவை.

பிரெஞ்சுக் கணிதமேதையும் தத்துவவாதியுமான பாஸ்கல் (Blaise Pascal) என்பவர் பல்சக்கரங்களால் இயங்கும் ஒரு பொறியைத் தமது 19 ஆம் வயதில், 1642 இல் கண்டுபிடித்தார். இதில் ஒவ்வொரு சக்கரத்திலும் 0 முதல் 9 வரை எழுதப்பட்ட பத்துப் பல்கள் இருந்தன. பலவித முயற்சிகள் எடுத்தும் இப்பொறியை இவரால் விற்பனை செய்ய முடியவில்லையாம். இவரைக் கௌரவிப்பதற்காகத்தான் ஒரு கணிப்பொறி மொழிக்கு இவரது பெயர் இடப்பட்டிருக்கிறது. கணிப்பொறி வருவதற்கு முன்பு வரை, பல் சக்கரங்களைக் கொண்டு கையினால் அல்லது மின்சாரத்தினால் இயங்கும் சிறு கணிப்பான்களை ஆராய்ச்சியாளர்களும், பொறி யாளர்களும் பயன்படுத்திவந்தனர். தட்டச்சுப் பொறியின் அளவில் பாதி இருக்கும் இப்பொறிகளை இன்றும் பல பொறியியற் கல்லூரிகளின் கிடங்குகளில் காணலாம்.

பாஸ்கலின் கணிப்பான், கூட்டலும், கழித்தலும் செய்யும்; அவற்றோடு பெருக்கல், வகுத்தலைச் சேர்த்தவர் லைப்னீஸ் (Leibniz).

பாபேஜ்-கணிப்பொறியின் தந்தை

அடுத்த மைல்கல்லாகக் குறிப்பிடப்படுபவர், இன்றைய கணிப்பொறியின் தந்தை என்று போற்றப்படும் பாபேஜ் (Charles Babbage 1792 - 1871) என்னும் பிரிட்டிஷ்காரர். இவர் ஒரு கணித மேதை; பொறியாளரும் கூட. இவர் முதலில் பல் சக்கரங்களால் இயங்கும் **டிபரன்ஸ் என்ஜின்** (Difference Engine) என்ற கணக்கிடும் பொறியை உருவாக்கினார். இதனை விரிவுபடுத்தி பெரிதாகச் செய்யத் திட்டமிட்டார். இத்திட்டத்தைச் செயலாக்கத் தேவையான பகுதிகள் முழுவதையும் இவரே வடிவமைக்க வேண்டியதாயிற்று. பொருளின்மை காரணமாக இவரால் அந்தப் புதிய இயந்திரத்தைச் செய்து முடிக்க முடியவில்லை.

கனவுகள்

1833 ஆம் ஆண்டுவாக்கில், பலவிதக் கணக்குகளைச் செய்யும்படி மாற்றிக் கொள்ளவல்ல ஆற்றல் மிக்க ஒரு புதிய கணிப்பொறியை வடிவமைக்கத் தொடங்கினார். அந்த வடிவமைப்பு காகித வ்டிவிலேயே முற்றுப்பெற வேண்டி வந்தது. ஏனென்றால், அந்த வடிவமைப்பைச் செயல்படுத்தும் பொறியியல் திறன் அன்று இல்லை; அவரிடம் பணமும் இல்லை. அவருடைய எண்ணங்களின் பிரதிபலிப்பு இன்றைய கணிப்பொறிகளில் பெரிதும் காணப்படுவதால்தான் அவரை இன்றைய கணிப்பொறியின் உண்மையான தந்தை என்று கூறுகின்றனர். தான் வடிவமைத்த கணிப்பொறிக்கு அவர் **அனலிடிகல் என்ஜின்** (Analytical Engine) என்று பெயரிட்டிருந்தார். இவருடைய கனவுகள் நனவாக நூறாண்டுகள் ஆகியிருக்கின்றன. நம் நலனுக்காக நூறாண்டுகளுக்கு முன்பே சிந்தித்த அறிஞர் அவர். பைரனின் மகளான **ஏடா லவ்வேஸ்** (Ada Lovelace) அனலிடிகல் என்ஜினைப் புரிந்துகொண்டு, அதனைச் செயல்படுத்தத் தேவையான **கட்டளை நிரல்களை** (Computer Programs) எழுதினார். அதனால்தான் அவரை முதல் **நிரலர்** (First Programmer) என்கின்றனர்.

யு.எஸ். மக்கள் தொகைக் கணிப்பு

அமெரிக்க ஐக்கியக் குடியரசில் பத்து ஆண்டுகளுக்கு ஒரு முறை மக்கள் தொகையைக் கணக்கிடுவது வழக்கம். பெரும் அளவில் தொகுக்கப்படும் தகவல்களை அலசி ஆராய்ந்து, பல பட்டியல்களைத் தயாரித்துக் கொடுக்கும் பணி அவ்வளவு எளிமையானதல்ல. 1880 ஆம் ஆண்டின் கணக்கெடுப்பினை ஆராய்ந்து முடிக்கச் சுமார் எட்டு ஆண்டுகள் ஆயினவாம். 1890 ஆம் ஆண்டுக் கணக்கெடுப்பும் விரைந்து வந்துவிட்டது. இதனை எவ்வாறு விரைவில் அலசி முடிப்பது என்பதை

ஹோல்ரித் (Herman Hollerith) என்பவர் ஆராய்ந்தார். அவர் எடுத்த முடிவின்படி, தகவல்கள் புதிய முறையில் கையாளப்பட்டன.

1. காகிதத்திற்குப் பதில் அட்டைகள்.
2. எழுத்துக்களுக்குப் பதிலாகத் துளைகளைப் பயன்படுத்தும் குறியீடு.
3. துளைகளைப் பார்த்து, அதன் மூலம் பொறுக்கி எடுக்கும் இயந் திரங்கள்.

இம்முறையில் 1890 ஆம் ஆண்டுக் கணக்கெடுப்பின் ஆய்வு மூன்றே வருடங்களில் முடிந்துவிட்டது. துளையிட்ட அட்டைகளை ஏற்கனவே ஜக்கார்ட் (Joseph Marie Jacquard) என்பவர் நெசவுத் தறிகளில் 1880களில் பயன்படுத்தி யிருக்கிறார். அந்த வழியைத் தகவல்களுக்கும் பயன்படுத்தலாம் என்று நிரூபித்தவர் ஹோல்ரித். இம்முறையின் வெற்றியைப் பார்த்துப் பல நாடுகளும், காப்பீட்டு நிறுவனங்களும் ஹோல்ரித்தின் பொறிகளைப் பயன்படுத்தத் தொடங்கினர். அதனால், ஹோல்ரித் Tabulating Machine Company என்ற நிறுவனத்தைத் தொடங்கினார். இதுதான் இன்று புகழ்பெற்ற IBM (International Business Machines Corporation) நிறுவனத்தின் ஒரு பகுதியாக மாறியது.

நினைவில் நிற்க

1. மணிச்சட்டம், பெருக்கல் குச்சிகள், நகரும் சட்டம் முதலியன தொடக்க காலக் கணிப்புக் கருவிகள்.

2. பாஸ்கலின் கணிப்பான், பாபேஜின் டிபரன்ஸ் என்ஜின், ஹோல்ரித்தின் பொறிகள் முதலியன கணிப்பொறி வரலாற்றின் முதல் கட்டம். இவை நகரும் பாகங்களையும் பல்சக்கரங்களையும் கொண்டு இயங்கிய பொறிகள்.

பயிற்சி வினாக்கள்

1. பாஸ்கல் தயாரித்த கணிப்பான்
   அ. சட்டங்களால் ஆனது ஆ. பல் சக்கரங்களால் ஆனது
   இ. மின்சாரத்தால் இயங்கியது் ஈ. மின்னணு இயந்திரம்

2. பாபேஜ் செய்து முடித்த பொறியின் பெயர்
   அ. அனலிடிகல் என்ஜின் ஆ. டிபரன்ஸ் என்ஜின்
   இ. எனியாக்

3. பாபேஜ் வடிவமைத்து, ஆனால் செய்து முடிக்காத கணிப்பொறி
   அ. டிபரன்ஸ் என்ஜின் ஆ. எனியாக்
   இ. மார்க்-1 ஈ. எட்சாக்

4. நவீன கணிப்பொறியின் தந்தை
   அ. பாஸ்கல் ஆ. ஹோல்ரித்
   இ. பாபேஜ் ஈ. ஃபான் நாய்மன்.

1.2 கணிப்பொறியின் அண்மைக்கால வரலாறு

இரண்டாம் உலகப்போர் தொடங்கியபோதும் அதற்கு முன்பும் (சுமார் 1930 முதல் 1945 வரை) இராணுவ நடவடிக்கைகளுக்காக, விஞ்ஞானக் கணிப்புகள் பல செய்ய வேண்டியிருந்தன. உதாரணமாக, எந்தச் சூழ்நிலைகளில், எந்தக் கோணங்களில் குண்டு வீசினால், எவ்வளவு தொலைவில் போய் அது விழும் என்பது போன்ற தகவல்கள் அடங்கிய பட்டியல் தயாரித்தல். மிகவும் நேரமெடுக்கும் இக்கணிப்புகளைச் செய்துமுடிக்கப் பொறிகளால்தான் முடியும் என உணர்ந்து அமெரிக்க, ஐரோப்பிய நாடுகள் இத்துறைக்குப் பெரிதும் ஊக்கமளித்தன. இராணுவம் தொடர்பானது என்பதனால் இத்துறையில் இந்த நாட்களில் அடைந்த முன்னேற்றங்கள் பல இன்னமும் இரகசியமாகவே உள்ளன.

எலெக்ட்ரோமெகானிகல் ரிலே

எலெக்ட்ரோமெகானிகல் ரிலே (Electro Mechanical Relay) எனப்படும் மின்சாரத்தால் நகரும் பாகங்களை ஆதாரமாகக் கொண்டு பெல் தொலைபேசி நிறுவனம் (Bell Telephone Laboratories) செய்து கொடுத்த கணிப்பொறிகள் யு.எஸ். படைகளுக்குப் பயன்பட்டன.

மார்க் 1

ஹார்வர்ட் பல்கலைக் கழகத்தில் ஐகென் (Howard Aiken) 1937-44 இல் கட்டிய கணிப்பொறியின் பெயர் மார்க் 1. இதுவும் எலெக்ட்ரோ மெகானிகல் ரிலே வகைதான். இது ஐ.பி.எம், யு.எஸ். கடற்படை உதவியுடன் உருவாகியது. இப் பொறியில் கணிதச் செயல்பாடுகளின் வரிசையை நம் கட்டுப்பாட்டிற்கிணங்க மாற்றிச் செயல்படுத்த முடிந்தது. பாபேஜின் கனவை நனவாக்கிய கணிப்பொறி இது என்பர். இதனை உருவாக்கிய ஐகெனுக்கு பாபேஜின் பணியைப் பற்றித் தெரியும். அதனைக் குறிப்பிட்டுக் கூறியுமிருக்கிறார். இதே காலத்தில் (1941) ஸ்யூஸ் (Zuse) என்னும் ஜெர்மானியரும் எலெக்ட்ரோ மெகானிகல் வகைப் பொறியைச் செய்திருந்தார்.

கணிப்பொறி - முதல் தலைமுறை

அசையும் பாகங்களை நீக்கிவிட்டு, அசையாத மின்னணுக் கருவியாக உருவெடுத்தபோதுதான் உண்மையான கணிப்பொறி பிறந்து அதன் முதல் தலைமுறை தொடங்கியது என்பர். இந்த முதல் தலைமுறை பயன்படுத்தியது மின்னணு வெற்றிடக் குழாய்களாகும் (Vacuum tubes).

அயோவா மாநிலக் கல்லூரியில், 1939 இல் அடனாஸோஃப் (Joseph Vincent Atanasoff) என்பவரும், அவர் மாணவர் பெர்ரி (Clifford Berry) என்பவரும்

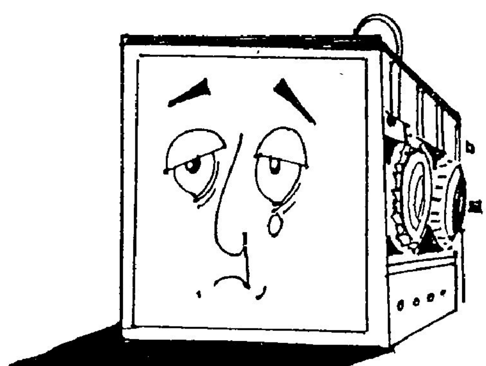
மின்னணுக் கணிப்பொறி (Electronic Computer) ஒன்றினைச் செய்து முடித்தனர். இதன் மற்றொரு சிறப்பு, இவர்கள் தசம எண்களுக்குப் பதிலாக **இருநிலை எண்களை** (Binary Numbers) முதன் முதலாகப் பயன்படுத்தியதாகும். பிரிட்டிஷ் உளவுத் துறையின் சங்கேத மொழிகளை உடைப்பதற்காக இந்தக் காலத்தில் பயன்படுத்திய கொலாஸி (Colossis) என்றவகை மின்னணுக் கணிப்பொறி பற்றி இன்னமும் அதிகம் தெரியவில்லை.

எனியாக் (ENIAC - Electronic Numerical Integrator and Calculator)

1943 முதல் *1946* வரை கட்டப்பட்ட இக்கணிப்பொறியை முதல் பெரிய மின்னணுக் கணிப்பொறி எனலாம். இதில் மின்னணு வெற்றிடக் குழாய்கள் பயன்படுத்தப்பட்டன. நகரும் பகுதிகள் இல்லாததால், இப்பொறியானது நகரும் பகுதிகள் கொண்ட பழைய பொறிகளைவிடச் சுமார் *1000* பங்கு வேகத்தில் செயல்பட முடிந்தது.

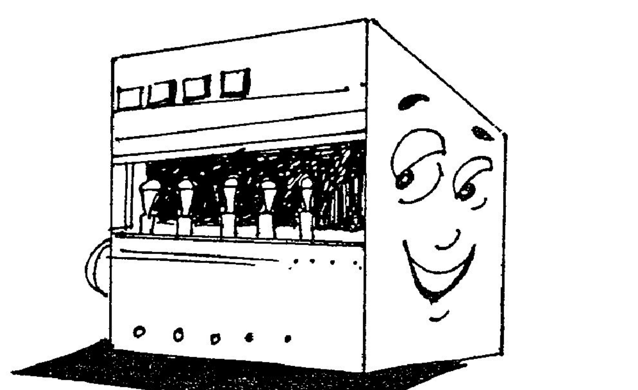
இதனை உருவாக்கிய மௌஷ்லி (John W Mauchly), எக்கெர்ட் (J Presper Eckert) என்பவர்களுள் முதலாமவர் ஏற்கனவே அயோவா மாநிலக் கல்லூரியில் அடனாஸோஃப் உருவாக்கிய மின்னணுக் கணிப்பொறியைப் பார்த்திருந்தார். அதிலிருந்து தெரிந்துகொண்டதில் எந்த அளவு தன் கணிப்பொறியில் பயன்படுத்தினார் என்பது தெரியவில்லை.

"ஓடி ஓடி வேலை செஞ்சு என்ன பயன்?
வேகம் போதாதுன்னுதான் சொல்றாங்க"



எலக்ட்ரோ மெகானிகல் வகை

"அம்மாடி! இப்படி நகராமல் வேலை செஞ்சா,
எவ்வளவு வேலை செய்ய முடியுது!"



மின்னணு வகை

இந்த எனியாக் பொறியின் அளவுகள், இன்று மேஜைக் கணிப்பொறிகளைப் பார்க்கும் நம்மை வியப்பிலாழ்த்தும். *18,000* வெற்றிடக் குழாய்கள், *10,000* தடுப்பான்கள் (resistors), *10,000* கெபாஸிடர்கள் (capacitors) முதலியவற்றை

உள்ளடக்கியது இப்பொறி. 15,000 சதுரஅடி பரப்பளவில், 30 டன் எடையைக் கொண்டதாகவும், 150 கிலோவாட் சக்தியை உட்கொள்வதாகவும் இருந்தது! இப்பொறி வினாடிக்கு 5,000 கூட்டல்கள் அல்லது 300 பெருக்கல்களைச் செய்ய வல்லது. சுமார் 10 வருடங்கள் உழைத்துவிட்டு ஓய்ந்தது.

கணக்குகளை விரைவாகப் போட்டு முடித்தாலும், இதனைப் பயன்படுத்துவது அவ்வளவு எளிதல்ல; இந்தப் பொறியைப் பற்றி நன்கு தெரிந்திருக்க வேண்டும். ஏனென்றால், நம் கணக்கிற்கேற்ப, சில இணைப்புகளையே மாற்றி அமைக்க வேண்டியிருக்கும்! இவ்வாறு செய்ய சில சமயங்களில் ஓரிரு நாள்கள் கூட ஆகிவிடும்.

ஃபான்நாய்மன் (John Von Neuman)

எனியாக் கணிப்பொறியை உருவாக்குவதில் பங்கேற்றிருந்த ஃபான் நாய்மன் ஒரு புதுக்கருத்தைக் கூறினார். அதுவரை, கணக்கிற்கான தகவல்களை மட்டும் கணிப்பொறிக்குள் வைத்திருந்தார்கள். அவற்றை என்ன செய்ய வேண்டும் என்ற கட்டளைகளை வெளியே இருந்து அனுப்பினார்கள் அல்லது அதையே மின் இணைப்புகளை மாற்றுவதன் மூலம் செய்தார்கள்.

கணக்கிற்கான தகவல்களைப் போலவே, **கட்டளைத் தொகுப்பையும் கணிப்பொறியின் நினைவகத்திலேயே வைத்துக்கொண்டு வேலை செய்யலாம்.** இதுவே ஃபான் நாய்மனின் புதுக்கருத்து. இதனால் இரு பெரும் நன்மைகள் விளைந்தன. ஒன்று - கட்டளைகளை உள்ளே அனுப்பும் நேரம் மிகவும் குறைந்தது. இரண்டு - தேவைக்கேற்பக் கட்டளைகளின் வரிசையை மிக எளிதில் மாற்றிக் கொள்ள முடிந்தது.

மேலே உள்ள கருத்தினை **கட்டளை நிரலை நினைவில் வைக்கும் முறை** (Stored Program Concept) என்பார்கள். இது கணிப்பொறி இயலில் மற்றுமொரு மைல்கல். ஃபான் நாய்மனின் இக்கருத்தைப் பயன்படுத்தி அமைத்த எட்வாக் (EDVAC - Electronic Discrete Variable Automatic Computer) என்பது தாமதமாக 1952 இல் தான் முடிந்தது. ஆனால் இதே முறையைப் பின்பற்றி வந்த முதல் கணிப்பொறி **எட்சாக்** (EDSAC - Electronic Delay Storage Automatic Calculator). இது 1949 இல் வெளிவந்தது. கேம்பிரிட்ஜ் பல்கலைக் கழகத்தில் உருவானது. இதைத் தொடர்ந்து பல பொறிகள் இதேபோல் வந்தன.

டிரான்ஸிஸ்டர் (Transistor)

தடையின்றி வேலை செய்யும் என்று நம்பமுடியாத வெற்றிடக் குழாயை விட்டுவிட்டு, நம்பகமான டிரான்ஸிஸ்டரைப் பயன்படுத்தத் தொடங்கியது ஒரு பெரும் மாற்றத்தின் முதல் படியாக அமைந்தது. இவ்வகையில் வெளிவந்த முதல் பொறி யூனிவாக் 1 (UNIVAC I - Universal Automatic Computer). வெற்றிடக் குழாய்களைப் பயன்படுத்திய கணிப்பொறிகளை முதல் தலைமுறை என்றும்,

டிரான்ஸிஸ்டர்களைப் பயன்படுத்தியவற்றை இரண்டாம் தலைமுறை என்றும் கூறுவார்கள். டிரான்ஸிஸ்டர்களைப் பயன்படுத்தியதால் கணிப்பொறியின் உருவம் சிறிதாயிற்று; வேகம் கூடியது; தேவைப்படும் மின்சக்தியின் அளவு குறைந்தது; நம்பகத்தன்மை அதிகரித்தது; விலை குறைந்தது. ஒவ்வொரு தலைமுறைக்கும் இம்மாதிரியான கூற்றுகள் பொருந்தும். IBM 1620 இரண்டாம் தலைமுறைக் கணிப் பொறி. இதன் நினைவகம் காந்த வளையங்களால் ஆனது (Magnetic Core Memory).

சில்லுகள் (Chips)

சில நூறு டிரான்ஸிஸ்டர்களை உள்ளடக்கிய ஒருங்கிணைந்த சுற்றுச் சில்லுகளைப் (Integrated Circuit Chips) பயன்படுத்திய கணிப்பொறிகள் மூன்றாம் தலைமுறையாகும். சில்லுகளின் தொடக்க காலம் இது. IBM 360 இந்தத் தலை முறையைச் சேர்ந்தது.

நான்காம் தலைமுறை

சில்லுகளைச் செய்யும் தொழில்நுட்பம் வளர்ந்து, பெருமளவு ஒருங் கிணைந்த சில்லுகள் (LSI Chips - Large Scale Integration Chips) வந்தன. ஒரு சில்லில் ஆயிரக்கணக்கான டிரான்ஸிஸ்டர்கள் அடங்கின. இவற்றைக் கொண்டு நான்காம் தலைமுறைக் கணிப்பொறிகள் உருவாயின. IBM 370 இந்தத் தலை முறைக் கணிப்பொறி.

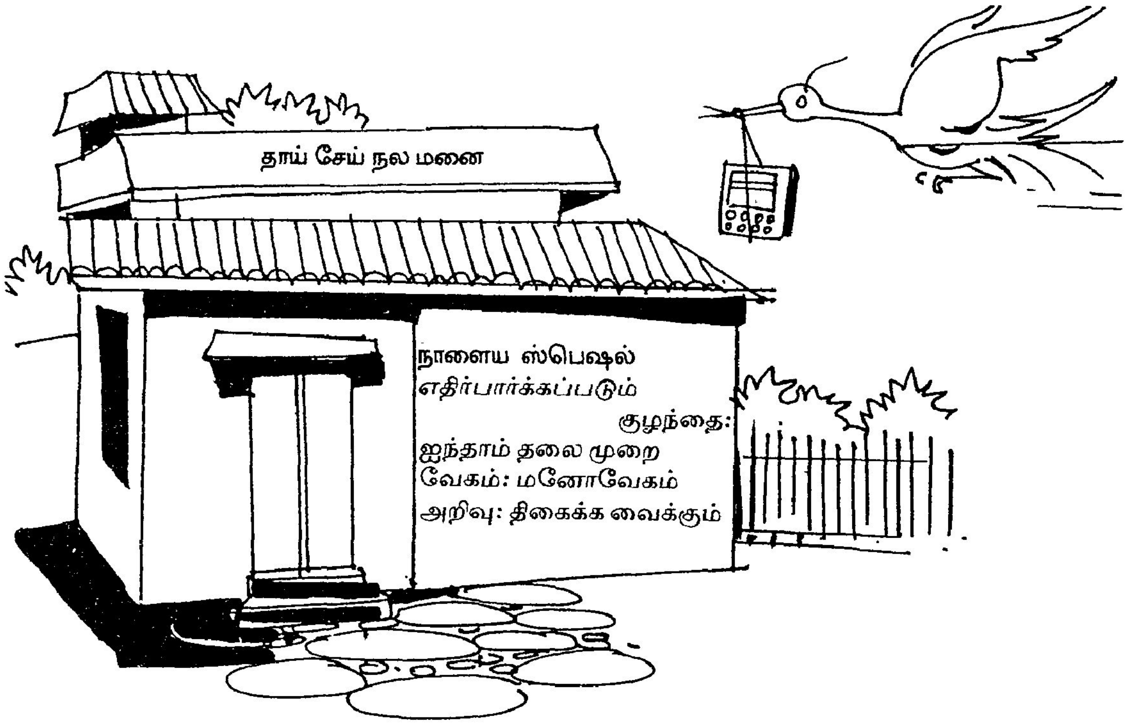
பெருமளவு ஒருங்கிணைப்பு வளர்ந்து, மிகப்பெருமளவு ஒருங்கிணைப் பானது (VLSI - Very Large Scale Integration). இத்தகைய சில்லுகள் பத்தாயிரத் திற்கும் மேற்பட்ட பாகங்களைத் தம்முள் கொண்டிருக்கும். சில்லுகள் தயாரிப் பதில் இன்டெல் (Intel) நிறுவனம் பெயர் பெற்றது. இதன் தயாரிப்புகளான 8088, 80826, 80836, 80486, பென்டியம் வகைச் சில்லுகள் இன்று பல **நுண் கணிப் பொறிகளின்** (Micro Computers) இதயமாக விளங்குகின்றன. பெரிய, மிகப் பெரிய அளவில் ஒருங்கிணைந்த சில்லுகளைப் பயன்படுத்தும் கணிப்பொறிகள் எல்லாம் நான்காம் தலைமுறையைச் சேர்ந்தவை.

ஐந்தாம் தலைமுறை

தாமே சிந்தித்து செயல்படத்தக்கபடி **செயற்கை அறிவுத்திறன்** (Artificial Intelligence) கொண்டதாக இருக்க வேண்டும் என்ற நோக்கத்துடன் ஐந்தாம் தலைமுறைப் பொறிகள் வடிவமைக்கப்பட்டு வருகின்றன. இன்னும் அவை வெளிவரவில்லை.

வளர்ச்சி

நாட்கள் செல்லச் செல்ல கணிப்பொறியின் ஆற்றல் அதிகரித்துக் கொண்டே வருகிறது. ஆற்றல் என்பதில் வேகம், உள் வெளி நினைவகங்களின் அளவு, சொல்



நீளம் முதலியவை அடங்கும். (இவை பற்றிப் பிறகு விரிவாகப் பார்ப்போம்). நுண்கணிப்பொறி வரத்தொடங்கிய காலத்தில் அவற்றின் நினைவகங்களின் அளவு சுமார் 10 KB. இது படிப்படியாக 32, 64, 256, 640, 1000, 2,000, 4,000, 16,000 என்று சில ஆண்டுகளுக்குள்ளேயே மகத்தான வளர்ச்சி பெற்றுவிட்டது. இதேபோல் வேகமும் 4.7 MHz, 8, 25, 33, 66, 90, 125, 200, 400 MHz என்று வளர்கிறது. சொல் நீளம் 8, 16, 32, 64 என்று வளர்ச்சி பெறுகிறது.

''தம்மின் தம்மக்கள் அறிவுடைமை மாநிலத்து மன்னுயிர்க் கெல்லாம் இனிது''
கணிப்பொறிக்கும் கூடத்தான்!

நினைவில் நிற்க

1. மின்சாரத்தினால் இயங்கி நகரும் பாகங்களைக் கொண்ட மார்க் 1, பாபேஜின் கனவை நனவாக்கிய கணிப்பொறி என்பார்கள்.

2. அடனாஸோஃப்பும், அவர் மாணவர் பெர்ரியும் முதல் மின்னணுக் கணிப்பொறியை உருவாக்கினர். இதில் இருநிலை எண்களை முதன் முதலாகப் பயன்படுத்தினர்.

3. எனியாக் என்பது முதல் பெரிய மின்னணுக் கணிப்பொறி. வெற்றிடக் குழாய்களால் இயங்கியது.

4. ஃபான் நாய்மனின் கட்டளை நிரலை நினைவில் வைக்கும் முறையைப் பயன்படுத்திய கணிப்பொறி எட்சாக்.

5. வெற்றிடக் குழாய்களுக்குப் பதிலாக டிரான்ஸிஸ்டர்களும், பிறகு சில்லு களும் பயன்படுத்தப்பட்டன.

6. செயற்கை அறிவுத்திறன் கொண்டதாக இருக்கும்படி ஐந்தாம் தலை முறைக் கணிப்பொறிகள் உருவாக்கப்பட்டு வருகின்றன.

பயிற்சி வினாக்கள்

1.மார்க் 1 என்பது இவ்வகை இயந்திரம்

அ. எலெக்ட்ரோ மெகானிகல் ரிலே ஆ. மின்னணு

இ. மெகானிகல் ஈ. சில்லுகள்

2. பாபேஜின் கனவை நனவாக்கிய கணிப்பொறி

அ. எட்னாக் ஆ. எட்சாக்

இ. எனியாக் ஈ. மார்க் 1

3. கட்டளை நிரலை நினைவில் வைக்கும் முறையை அறிமுகப்படுத்தியவர்

அ. பாபேஜ் ஆ. ஃபான்நாய்மன்

இ. பாஸ்கல் ஈ. அடனாஸோஃப்

4.வெற்றிடக் குழாயைப் பயன்படுத்தி வெற்றிகரமாகச் செயல்பட்ட ஒரு பெரிய புகழ் பெற்ற கணிப்பொறி.

அ. எட்சாக் ஆ. எட்வாக்

இ. எனியாக் ஈ. ஆப்பிள்

5. கணிப்பொறியின் தலைமுறைகளை நிர்ணயிக்கும் முக்கிய அம்சங்கள் எவை?

1.3 கணிப்பொறியின் வகைகள்

கணிப்பொறிகளின் அளவையும், சக்தியையும் வைத்து அவற்றை **உயர்** (Super), ***பிரதான*** (Mainframe), ***சிறு*** (Mini), ***நுண்*** (Micro) கணிப்பொறிகள் என்று தரம் பிரிக்கலாம். அவற்றைப் பற்றி இந்தப் பாடத்தில் காண்போம்.

பிரதான கணிப்பொறிகள் (Mainframe Computers)

Tabulating Machine Company என்று ஹோல்ரித் தொடங்கிய நிறுவனம், பிறகு மற்ற மூன்று சிறு நிறுவனங்களுடன் இணைந்து Computing Tabulating Recording Company என்று ஆனது. 1924 இல் இது International Business Machines Corporation என்று பெயர் மாற்றம் பெற்றது. 1950 வரை துளையிட்ட அட்டை களைப் பயன்படுத்தும் பொறிகளில்தான் கவனம் செலுத்தி வந்தது. 1953இல் தொடங்கி இன்றுவரை பல புதிய புதிய வகைகளை (models) அறிமுகப்படுத்திக் கொண்டே இருக்கிறது. வகை 650, 701, ..., 706, 709 வரை வெற்றிடக் குழாய் வகைகள். 7090, 7040, 7044 போன்றவை டிரான்ஸிஸ்டர் பயன்படுத்தியவை. இவ்வகையில் புகழ் பெற்றது 1620 வகை. அடுத்து வந்த 360, 370 வகைகள் மிகவும் சக்தி வாய்ந்த, உலகம் முழுவதும் விற்பனையான வகைகள்.

இந்த IBM 360, 370 போன்ற கணிப்பொறிகளையெல்லாம் பிரதான வகை (Main frame) கணிப்பொறிகள் என்பார்கள். இப்பொறிகளெல்லாம் **துளை அட்டைப் படிப்பான்கள்** (Card readers), **காந்த நாடாக் கருவிகள்** (Tape units), **அச்சுப் பொறிகள்** (Printers) போன்று பலவித உள்/வெளியீட்டுச் சாதனங்களைக் கையாளும் திறம் பெற்றதால் இப்பெயர் பெற்றன. இவற்றின் விலை பல இலட்சங்களில் அல்லது சில கோடிகளில் இருக்கும். எல்லோராலும் வாங்க முடியாது.

சென்னை கிண்டி பொறியியற் கல்லூரியில் 1963ஆம் ஆண்டு வாங்கிய IBM 1620 அன்று தென்னிந்தியாவிலேயே ஒரு முன்னோடியாக விளங்கியது. எழுபதுகளின் தொடக்கத்தில் சென்னையிலுள்ள இந்திய தொழில்நுட்பக் கழகம் (IIT) வாங்கிய IBM 370 பொறியும் குறிப்பிடத்தக்கதாகும். அறுபது, எழுபதாம் ஆண்டுகளில் இந்தியாவில் ஒரு சில கணிப்பொறிகளே இருந்தன.

சிறு கணிப்பொறிகள் (Mini Computers)

பிரதான வகைகளின் விலை மிக அதிகமாக இருந்ததால் சற்று விலை குறைந்ததாகவும், சற்று திறமை குறைவாகவும் உள்ள கணிப்பொறிகளுக்குத் தேவை ஏற்பட்டது. இந்த வகையில் புகழ் பெற்றது PDP என்னும் வரிசையில் வந்த பொறிகள். இவை Digital Equipment Corporation ஆல் உருவாக்கப்பட்டவை.

அறுபதுகளில் தொடங்கிய இவ்வகைக் கணிப்பொறிகள் சிறு கணிப்பொறிகள் எனப்பட்டன. இவற்றுக்கு அடுத்து வந்த Vax, Micro vax என்பவை திறனில் மிகுந்தும், விலையில் குறைந்தும் புகழ்பெற்ற சிறுபொறிகள்.

நுண்கணிப்பொறிகள் (Micro Computers)

கணிப்பொறி ஓர் அதிசயமான பொருள் என்ற நிலையில் இருந்து, இன்று எல்லோருக்கும், எல்லா இடங்களிலும் பயன்படும் ஓர் அவசியப் பொருள் என்ற நிலைக்கு வரக் காரணமே நுண்கணிப் பொறிகள்தான். இவை மேஜை மேல் வைக்கத்தக்க அளவில் இருக்கும். கையில் எடுத்துச் செல்லும் அளவிலும் இன்று கிடைக்கின்றன. இவை மடிமேல் வைத்துக் கொள்ளும் வகையும் (Lap Top Computer), நோட்டுப் புத்தக (Note Book Computer) அளவு வகையும் ஆகும்.

IBM இனம்

நுண்கணிப்பொறிகளில் மிகவும் பரவலாகக் காணப்படுவது IBM தயாரிக்கும் பொறிகளும், அதேபோல் வேலைசெய்யும், ஆனால் மற்றவர்கள் தயாரிக்கும் IBM இனமும்தான் (IBM Clones). இதற்கு முக்கிய காரணம் இவ்வினத்தில் ஒரு கணிப் பொறியில் செயல்படும் எந்த **மென்பொருளும்** (Software) அந்த இனத்தின் எல்லாக் கணிப்பொறிகளிலும் செயல்படும். அதனால் ஏராளமான மென்பொருட்கள் இந்த இனத்திற்குக் கிடைக்கின்றன.

ஆப்பிள்

ஆப்பிள் (Apple) என்னும் வகைக் கணிப்பொறிகள் ஒரு தனி இனத்தைச் சேர்ந்தவை. இந்தவகை, மாணவர்களைக் கவர்வதாகவும், படங்களைக் கையாள்வதில் திறமை பெற்றதாகவும் உள்ளது. ஆப்பிள்தான் நுண்கணிப் பொறிகளின் தொடக்கமாக விளங்கியது. இதை 1975 இல் வடிவமைத்தவர் ஸ்டீவ் வாஸ்னியாக் (Steve Vasniak) என்னும் 25 வயது இளைஞர். ஆப்பிள் கணிப்பொறிகள் இன்று **மேஜைப் பதிப்பில்** (Desk Top Publishing) முன்னணியில் உள்ளன.

அடாரி (Atari) என்னும் மற்றொரு வகைக் கணிப்பொறிகள் விளையாட்டுகளுக்கு முக்கியத்துவம் தரும் வகையில் அமைக்கப்பட்டவை.

நுண்கணிப்பொறிகள் விலை குறைந்தவை. அவற்றின் திறன் கூடிக் கொண்டே போகிறது. அதனால் இன்று நம் நாட்டில் பல சிறிய நிறுவனங்கள் கூட கணிப்பொறிகளைப் பயன்படுத்தத் தொடங்கிவிட்டன.

நுண்கணிப்பொறிகளை **சொந்தக் கணிப்பொறி** (Personal Computers) என்றும் கூறுவார்கள்.

உயர் கணிப்பொறிகள் (Super Computers)

சிறிய மற்றும் நுண் கணிப்பொறிகளின் தேவை அதிகரித்தது போலவே மிகச் சிக்கலான பெரிய கணக்குகளைத் தீர்ப்பதற்கு, மிக விரைவாகச் செயல்படும், பெரிய நினைவகங்களுள்ள, திறமை மிகுந்த கணிப்பொறிகளுக்கும் தேவை அதிகரித்தது. இவற்றை வடிவமைப்பதும் கடினம். விற்பனையாகும் எண்ணிக்கையும் குறைவு. அதனால் விலை மிக அதிகம். அணுசக்தி, பருவ நிலைக்கணிப்பு போன்ற துறைகளுக்கு இப்பொறிகள் மிக அவசியம். கிரே (CRAY) என்ற வகைப் பொறிகள் இந்த உயர்வகையைச் சேர்ந்தவை. இந்தியாவில் பரம் (PARAM) போன்ற உயர்வகைக் கணிப்பொறிகளை உற்பத்தி செய்யத் தொடங்கியுள்ளோம். ஜப்பானும் இத்துறையில் முன்னணியில் உள்ளது.

நினைவில் நிற்க

நாம் இதுவரை பார்த்த கணிப்பொறி வரலாற்றின் சாரம் சில பட்டியல்களாகக் கீழே கொடுக்கப்பட்டுள்ளது.

பட்டியல் 1.1 கணிப்பொறி வரலாற்றில் இடம்பெற்ற சிலர்

அறிஞர் பெயர்	கண்டுபிடிப்பு	ஆண்டு
1. பாஸ்கல்	பல்சக்கரக் கணிப்பான்	1642
2. பாபேஜ்	டிபரன்ஸ் என்ஜின், அனலிடிகல் என்ஜின் (அவர் முடிக்கவில்லை)	1792-1871
3. ஹோல்ரித்	துளை அட்டைகள்	1890
4. அடனாஸோஃப், பெர்ரி	முதன் மின்னணுக் கணிப்பொறி, இருநிலை எண்களைப் பயன்படுத்துதல்	1939
5. ஐகென்	மார்க் 1	1944
6. மௌஷ்லி, எகெர்ட்	முதல் பெரிய மின்னணுக் கணிப்பொறி எனியாக்	1946
7. ஃபான் நாய்மன்	கட்டளை நிரலை நினைவில் வைத்தல்	1949
8. ஸ்டீவ் வாஸ்னியாக்	ஆப்பிள்-நுண்கணிப்பொறி	1975

பட்டியல் 1-2 கணிப்பொறியின் தலைமுறைகள்

	முதல் தலைமுறை	இரண்டாம் தலைமுறை	மூன்றாம் தலைமுறை	நான்காம் தலைமுறை
1. காலம்	1946-57	1958-64	1965-71	1972 -
2. மையச் செயலகத் தொழில்நுட்பம்	வெற்றிடக் குழாய்	டிரான்ஸிஸ்டர்	சிறு, நடுத்தர அளவில் ஒருங்கிணைந்த சில்லுகள்	பெரும், மிகப் பெரும் அளவில் ஒருங்கிணைந்த சில்லுகள்
3. நினைவகம்	எலெக்ட்ரோ ஸ்டாடிக்குழாய்	காந்த வளையம்	சில்லுகள்	சில்லுகள்
4. முதல் சில கணிப்பொறிகள்	எனியாக்	IBM 1620	IBM 360 PDP 8	IBM 370

பட்டியல் 1-3 இன்றைய கணிப்பொறி வகைகள்

	உயர்வகை	பிரதான வகை	சிறுவகை	நுண்வகை
1. விலை (ரூபாயில்)	சில கோடிகள், பல லட்சங்கள்	சில கோடிகள், பல லட்சங்கள்	சில லட்சங்கள்	சில ஆயிரங்கள்
2. எடுத்துக் காட்டு	CRAY PARAM	IBM 370	PDP-11 Micro Vax	IBM PC, XT, AT, PS-2, Sun Workstations, IBM Clones, Apple, Atari

பயிற்சி வினாக்கள்

1. பிரதான கணிப்பொறி வகையைச் சேர்ந்தது

அ. ஆப்பிள் ஆ. PS-2
இ. IBM 370 ஈ. PDP 11

2. சொந்தக் கணிப்பொறிக்குக் காரணகர்த்தா

அ. ஃபான்நாய்மன் ஆ. ஸ்டீவ் வாஸ்னியாக்
இ. மௌஷ்லி ஈ. பாஸ்கல்

3. இவர் பெயரில் ஒரு கணிப்பொறி மொழி உள்ளது.

அ. அரிஸ்டாடில் ஆ. ஹோல்ரித்
இ. பாபேஜ் ஈ. பாஸ்கல்

4. முதல் தலைமுறைக் கணிப்பொறியின் தொழில்நுட்பம்

அ. எலெக்ட்ரோ மெகானிக்கல் ரிலே ஆ. டிரான்ஸிஸ்டர்
இ. சில்லுகள் ஈ. வெற்றிடக்குழாய்

5. ஐந்தாம் தலைமுறைக் கணிப்பொறியின் முக்கியக் குறிக்கோள்

அ. பலரும் பயன்படுத்துவது ஆ. செயற்கை அறிவு
இ. பேசும் கணிப்பொறி ஈ. தீப்பெட்டி அளவு

6. ஒரு தலைமுறையைக் காட்டிலும் அடுத்த தலைமுறைக் கணிப்பொறி

அ. விலை அதிகமாகும் ஆ. நினைவகம் சிறியதாகும்
இ. விரைவாகச் செயல்படும் ஈ. உள்ளீட்டகம் மாறும்.

7. ஆப்பிள் என்பது

அ. சொந்தக் கணிப்பொறி ஆ. ஐந்தாம் தலைமுறைக் கணிப்பொறி
இ. உயர் கணிப்பொறி ஈ. முதல் மின்னணுக் கணிப்பொறி

8. கணிப்பொறித் தலைமுறைகள் என்றால் என்ன? அவற்றுக்குள் என்ன வித்தியாசம் ?

9. கணிப்பொறி வகைகள் எவை? ஏன் அவ்வாறு பிரிக்கப்படுகின்றன?

1.4 இலக்க வகையும் ஒப்புமை வகையும்

கணிப்பொறியின் இயங்கும் தத்துவத்தை வைத்து அவற்றை இருவகை களாகப் பிரிக்கலாம். அவை இலக்க வகை (Digital), ஒப்புமை வகை (Analog) எனப்படும்.

மின்விசிறியின் வேகக் கட்டுப்படுத்தியைப் பார்த்திருக்கிறீர்கள். சாதாரண மாக அதில், நான்கு அல்லது ஆறு நிலைகள் இருக்கும். அவை 0 முதல் 3 அல்லது 5 வரை உள்ள வேகங்களைக் குறிக்கும். இங்கு மாற்றம் படிப்படியாக இருக்கும். ஒரு நிலையை விட்டால் அடுத்த நிலைக்குத்தான் செல்ல வேண்டும். நடுவில் நிற்க முடியாது. இப்போது புதிதாக வரும் வேகக் கட்டுப்படுத்திகளைப் பார்த்திருக்கிறீர் களா? இதில் 0 முதல் முழு வேகம் வரை எங்கு வேண்டுமானாலும் வேகத்தை வைத்துக்கொள்ளலாம். இடைவெளி இல்லாமல். தொடர்ச்சியாக வேகத்தை மாற்ற முடியும்.

படிப்படியாக மாறுவது என்பதும், **தொடர்ச்சியாக மாறுவது** என்பதும் அடிப்படையில் வித்தியாசமான இரண்டு தத்துவங்கள். அவற்றின் குணங்களும் பயன்களும் வெவ்வேறானவை.

ஒப்புமைக் கணிப்பொறி

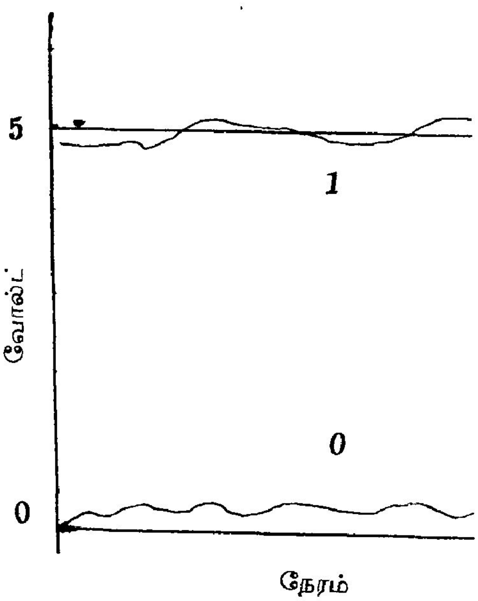
எந்தக் கணக்கிற்கும் ஈடாக, அந்தக் கணக்கிற்கு ஒப்பாக, ஒரு மின்சாரச் சுற்றை அமைக்க முடியும். உள்ளீடுகளும் வெளியீடுகளும் தொடர்ச்சியானவையாக இருந்தால் இம்முறையைப் பயன்படுத்தலாம். கணக்கில் நாம் கொடுக்கும் சில மதிப்புகளுக்கு ஏற்றபடி விடையும் மாறும். அதேபோல் இந்தச் சுற்றுகளில் உள்ளீட்டைக் குறிக்கும் இடங்களில் செல்லும் மின்சாரத்தின் அளவை மாற்ற மாற்ற, வெளியீட்டைக் குறிக்கும் இடங்களில் செல்லும் மின்சாரத்தின் அளவும் மாறும். இந்த அளவுகளையே விடையாகக் கொள்ள வேண்டும்.

இந்தச் சுற்றுகளில் பயன்படுத்தப்படும் பொருள்களின் தன்மைக்கேற்ப, வரும் விடையும் சரியான விடையிலிருந்து சற்று மாறுபடலாம். அதனால் துல்லியமாகக் கணக்கிட முடியாது. இது இவ்வகைக் கணிப்பொறிகளின் முக்கிய குறைபாடு. அடுத்த குறைபாடு, கொடுக்கும் கணக்கிற்கு ஏற்ப சுற்றுகளை மாற்றி அமைக்க வேண்டும். இதற்கு நேரம் அதிகம் பிடிக்கும். இதனால்தான் ஒப்புமைக் கணிப்பொறிகள் பிரபலமடையவில்லை.

இலக்கவகைக் கணிப்பொறி

மின்சார அளவுகளை, உதாரணமாக வோல்டேஜை, துல்லியமாக அளக்க முடியாது; அதனுடன், பயன்படுத்தும் பொருளைப் பொறுத்து இந்த மதிப்புகள் சற்று முன்பின் இருக்கலாம்; இதனால், வரும் விடையின் நுட்பம் பாதிக்கப்படும்

என்று பார்த்தோம். இந்தக் குறையைக் களைவதுதான் இலக்கவகையின் நோக்கம். இந்த வகைக் கணிப்பொறிகளில், ஓர் இடத்தில் ஓடும் மின்சாரத்தின் அளவு சுமார் 5 வோல்ட் என்றால் அது 1 என்பதையும், சுமார் 0 வோல்ட் என்றால் அது 0 என்பதையும் குறிப்பாக எடுத்துக்கொள்ளப்படும். சுமார் 5 வோல்ட் என்பதை, 4.5 அல்லது அதற்கு மேலும் என்றும், சுமார் 0 வோல்ட் என்பதை 0.5 அல்லது அதற்குக் கீழும் என்றும் வைத்துக் கொள்ளலாம். இதனால் நாம் மேலே பார்த்தபடி ஏற்படும் தவறுகளை முற்றிலும் நீக்கிவிடுகிறோம். (படம் 1.4).



படம் 1 - 4 பிழை நீக்கம்



"என்ன கண்ணா இது! வசசா குடுமி
இல்லாட்டா மொட்டைன்னு அடம் பிடிக்கிறியே!"

தற்போது வரும் புதிய கணிப்பொறிகளில் 5 வோல்டுகளுக்குப் பதிலாக 3.3 வோல்டுகளைப் பயன்படுத்துகின்றனர். இதனால் தேவைப்படும் மின்சாரத்தின் அளவு குறைகிறது.

இந்த முறையில் ஓர் இடத்தில் 0 என்று இருக்கும்; அல்லது 1 என்று அடுத்த நிலையில் இருக்கும். அதனால் இவ்வகைப் பொறிகளைக் கொண்டு கணக்கிடும்போது, ஒரு மதிப்பினைக் கணக்கிட்டால் அது படிப்படியாக மாறும். தொடர்ச்சியாக மாறாது. நாம் செய்யும் பெரும்பாலான கணக்குகளுக்குச்

தொடர்ச்சியான விடை தேவையில்லை. அதனால்தான் தேவையான அளவுக்குத் துல்லியமாகக் கணக்கிடும் இலக்கவகைக்-கணிப்பொறிகள் பிரபலமடைந்துள்ளன. இவ்வகையின் வெற்றிக்கு இன்னும் சில காரணங்களும் உண்டு. அவை,

1. பொறியில் எந்தவித மாற்றமும் செய்யாமலேயே, தேவையான எல்லாக் கணக்குகளுக்கும் விடை காணலாம்.

2. கணக்கிற்கு ஏற்றாற்போல், விடைகாணும் வழியை, கட்டளை நிரலாக உள்ளீடு செய்ய வேண்டும்.

3. உள்ளீடு செய்யப் பலவகையான உத்திகளையும், பொறிகளையும் பயன்படுத்தலாம்.

4. தேவையான வடிவங்களில், பொறிகளில் விடையை வெளியீடு செய்யலாம்.

5. இப்பொறிகளைப பயன்படுத்த, இவற்றைப் பற்றிய அறிவு அதிகம் தேவையில்லை.

6. ஒப்புமைக் கணிப்பொறிகளால் செய்ய முடியாத பலவகைத் தகவல் அலசல்களை இவ்வகையால் செய்ய முடியும்.

கலப்பின வகை (Hybrid Computers)

ஒப்புமை மற்றும் இலக்க வகைகளைத் தேவைக்கேற்றபடி இணைத்து உருவாக்கப்படும் பொறிகள் கலப்பின வகையைச் சேரும்.

நினைவில் நிற்க

1. ஒப்புமைக் கணிப்பொறி, தொடர்ந்து மாறும் மதிப்புகளைப் பயன்படுத்தும் கணிப்புகளைச் செய்யப் பயன்படும்.

2. இப்பொறியைக் கணக்கிற்குத் தகுந்தாற்போல் மாற்றி அமைக்க வேண்டும்.

3. இதன் பயன்பாடு ஒரு குறுகிய எல்லைக்கு உட்பட்டது.

4. இலக்கவகைக் கணிப்பொறியில் சுமார் 5 வோல்ட்டுகளை 1 என்றும், சுமார் 0 வோல்ட்டை 0 என்றும் எடுத்துக் கொள்வதால் துல்லியமாகக் கணக்கிட முடியும்.

5. இவ்வகையில் எல்லாவிதமான கணக்குகளுக்கும் எளிதில் விடை காணமுடியும்.

பயிற்சி வினாக்கள்

1. படிப்படியாக மாறுவது, தொடர்ச்சியாக மாறுவது என்ற தத்துவங்களை எடுத்துக்காட்டுகளுடன் விளக்கு

2. இலக்க வகைக் கணிப்பொறிகள் துல்லியமாகக் கணக்கிட உதவும் உத்தியை விளக்கு.

3. இன்றைய மேஜைக் கணிப்பொறி ஒரு

அ. ஒப்புமைக் கணிப்பொறி ஆ. கலப்பினக் கணிப்பொறி
இ. உயர் கணிப்பொறி ஈ. இலக்கவகைக் கணிப்பொறி

4. இலீக்கவகைக் கணிப்பொறியில் ஒரு கணக்கிற்கு விடைகாண

அ. கணக்கினைப் பொறியில் செலுத்த வேண்டும்.
ஆ. சுற்றுகளின் இணைப்பை மாற்ற வேண்டும்.
இ. விடையை எண்களாக மாற்ற வேண்டும்.
ஈ. கட்டளை நிரல் எழுத வேண்டும்.

5. இலக்கவகை ஒப்புமைக் கணிப்பொறிகளை ஒப்பிடும்போது

அ. ஒப்புமைக் கணிப்பொறி மிகவும் திறம் வாய்ந்தது.
ஆ. இரண்டுமே ஒரே திறம் வாய்ந்தவை.
இ. இலக்கவகைக் கணிப்பொறி மிகவும் திறம் வாய்ந்தது.

1.5 இருநிலை எண்கள்

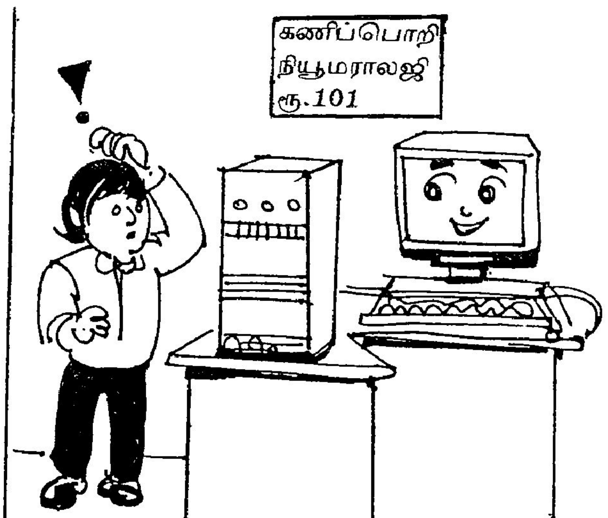
கணிப்பொறியானது, நகரும் பாகங்களை விட்டுவிட்டு, மின்னணு இயலைப் பயன்படுத்தத் தொடங்கியபோது, தசம முறையை விட்டுவிட்டு **இருநிலை முறைக்கு** (Binary System) மாறியது. ஏனென்றால் மின்சாரம் ஓரிடத்தில் பாய்கின்றதா (அதாவது சுமார் 5 வோல்ட்டா), இல்லையா என்பது இரு நிலைகளைக் கொண்டது. இம்முறையையும், இதனைக் கணிப்பொறிகளில் எவ்வாறு பயன் படுத்துகிறோம் என்பதையும் இப்பாடத்தில் தெரிந்துகொள்வோம்.

தசம எண்கள்

நமக்குப் பழக்கமான தசம எண்களைப்பற்றி முதலில் தெரிந்து கொள்வது அவசியம். தசமம் என்றால், பத்தை அடிப்படையாகக் கொண்டது என்று பொருள் படும். இந்த முறையில் 0, 1, 2,...., 9 என்று பத்து குறியீடுகள் உள்ளன. இவற்றையும் ஒரு புள்ளியையும் (தசமப் புள்ளி) வைத்து, பல எண்களையும் எழுதுகிறோம். ஓர் எண்ணை எழுதும்போது, அதில் உள்ள ஓர் இலக்கத்தின் மதிப்பு, அதனுடைய இடத்தைப் பொறுத்தது. இதனை **இடம் சார்ந்த குறியீடு** (Positional notation) என்பர். 127.24 என்பது, $(7 \times 10^0) + (2 \times 10^1) + (1 \times 10^2) + (2 \times 10^{-1}) + (4 \times 10^{-2})$ என்ற எண்ணைக் குறிக்கும். $10^0 = 1$ என்று எடுத்துக்கொள்ள வேண்டும். இந்தத் தசம முறை நம் கைகளில் பத்து விரல்கள் இருப்பதால் உருவான முறை. நம் கைகளில் மொத்தம் இரண்டே விரல்கள் இருந்தால் என்ன செய்திருப்போம்? யோசித்துப் பாருங்கள்.



"உலகில் மொத்தம் 9 வகை மனிதர்கள். உங்கள் பிறந்த தேதியின் கூட்டுத்தொகை 5. அதனால் ஐந்தாம் வகை. மற்றவர்களிடம்...."



"உலகில் மொத்தம் 1 வகை மனிதர்கள்; உங்கள் பிறந்த தேதியின் கூட்டுத்தொகை 1. அதனால் முதல் வகை. மிகுந்த...."

இருநிலை எண்கள்

இரண்டே விரல்கள் என்றால் மொத்தம் இரண்டே குறியீடுகள்தான்! அவற்றை 0,1 என்போம். எல்லா எண்களையும் இந்த இரண்டே குறியீடுகளால் குறிப்பிட முடியுமா? முடியும். இடம் சார்ந்த குறியீட்டின்படி, 10^0, 10^1, 10^2, 10^{-1}, 10^{-2},.... என்று 10 பயன்படும் இடங்களிலெல்லாம் 2 ஐப் பயன்படுத்த வேண்டும். அவவளவுதான். உதாரணமாக, 1101 என்பது (வலமிருந்து இடமாகக் கணக்கிடுவோம்), $(1 \times 2^0) + (0 \times 2^2) + (1 \times 2^3) + (1 \times 2^3) = 1 + 0 + 4 + 8 = 13$.

0 முதல் 19 வரை உள்ள எண்களை இருநிலை எண்களாக இவ்வாறு எழுத வேண்டும்.

0 - 0	4 - 100	8 - 1000	12 - 1100	16 - 10000
1 - 1	5 - 101	9 - 1001	13 - 1101	17 - 10001
2 - 10	6 - 110	10 - 1010	14 - 1110	18 - 10010
3 - 11	7 - 111	11 - 1011	15 - 1111	19 - 10011

இதேபோல் 10.101 என்றால் அது

$(0 \times 2^0) + (1 \times 2^1) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3}) = 2 + .5 + .125$
= 2.625 என்ற தசம எண்ணைக் குறிக்கும்.

தசம எண் பத்தை அடிப்படையாகக் கொண்டது. அதேபோல் இருநிலை எண் இரண்டினை அடிப்படையாகக் கொண்டது. இதேபோல் எட்டினை அடிப்படையாகக் கொண்ட **எண்ணிலை** (எட்டு நிலை = எண்ணிலை) எண்களையும் (Octal Numbers) பதினாறை அடிப்படையாகக் கொண்ட **பதினாறுநிலை** எண்களையும் கூடக் (Hexadecimal Numbers) கணிப்பொறியில் பயன்படுத்துகிறோம். 8,16 என்பதெல்லாம் இரண்டின் அடுக்குகள் ($2^3 = 8$, $2^4 = 16$) என்பதாலேயே அவை பயன்படுகின்றன. வெவ்வேறு அடிப்படையில் எண்களைக் குறிப்பிடும்போது அடிப்படை எண் எது என்று சந்தேகமின்றித் தெரிவதற்காக, அந்த அடிப்படையை ஒரு கீழ்க்குறியீடாகக் (Suffix) காண்பிக்கலாம். 110_2 என்பது இருநிலை எண். 65_8 என்பது எண்ணிலை எண்.

தசமத்திலிருந்து இருநிலை முறைக்கு

ஒரு இருநிலை எண்ணைத் தசம எண்ணாக மாற்றும் முறையை மேலே பார்த்தோம். இப்போது தசம எண்ணுக்குச் சமமான இருநிலை எண்ணைக் கண்டுபிடிக்கும் விதத்தைப் பார்ப்போம்.

முதலில் ஒரு முழு தசம எண்ணை எடுத்துக் கொள்வோம். இதை, ஈவு சுழியாகும் வரை, இரண்டால் வகுத்துக்கொண்டே வர வேண்டும். வரும் மீதிகளை வலமிருந்து இடமாக எழுதினால், தேவையான இருநிலை எண் கிடைக்கும். உதாரணம்.

2	13	மீதி
2	6	1
2	3	0
2	1	1
	0	1

13 = 1101

ஒரு தசம பின்னத்தை எடுத்துக் கொள்வோம். இதை இரண்டால் பெருக்க வேண்டும். கிடைக்கும் விடையின் பின்னப் பகுதியை மீண்டும் இரண்டால் பெருக்க வேண்டும். இவ்வாறு மீண்டும் மீண்டும் செய்ய வேண்டும். பின்னம் 0 என்று வந்தால் நிறுத்திவிடலாம். இல்லையென்றால், ஒரே பின்னம் திரும்பி வந்தாலும் நிறுத்தி விடலாம்; ஏனென்றால் இங்கு ஒரே இலக்கங்கள் திரும்பத் திரும்ப வரும். பெருக்கி வரும் விடைகளின் முழு எண்களை, இருநிலைப் புள்ளிக்கு (Binary Point) வலது புறம் வரிசையாக எழுதினால் தேவையான இருநிலை எண் கிடைக்கும். உதாணமாக,

0.6875 X 2 = 1.375	.2 X 2 = 0.4
.375 X 2 = 0.75	.4 X 2 = 0.8
.75 X 2 = 1.5	.8 X 2 = 1.6
.5 X 2 = 1.0	.6 X 2 = 1.2
.0 X 2 = 0	.2 X 2 = 0.4
0.6875 = 0.1011	$.2_{10}$ = 0.0011 0011....

ஓர் எண்ணில் முழு எண், பின்னம் என்னும் இரண்டுமே இருந்தால், அவற்றை இரண்டாகப் பிரித்து, மேலே உள்ளது போல் தனித்தனியாக மாற்றி, பிறகு ஒன்று சேர்க்க வேண்டும்.

ஒரு குறிப்பு

தசம எண்களை இருநிலை எண்களாக மாற்றும் பொழுது, கிடைக்கும் இலக்கங்களை இடமிருந்து வலமாக எழுதுவதா அல்லது வலமிருந்து இடமாக எழுதுவதா என்ற சந்தேகம் எழும். **கிடைக்கும் இலக்கங்களில் முதல் இலக்கம் புள்ளிக்கு அருகில் இருக்கும்** - இதை மனத்தில் வைத்துக்கொண்டால் போதும். சரியாக வரும்.

நினைவில் நிற்க

1. இரண்டின் அடிப்படையில் எண்களை எழுதலாம். அதற்கு 0,1 என்ற இரண்டு குறியீடுகள் போதும். இடம் சார்ந்த குறியீட்டு முறையைப் பயன்படுத்த வேண்டும்.

2. இருநிலை எண்களைத் தசம எண்களாகவும், தசம எண்களை இருநிலை எண்களாகவும் மாற்றலாம்.

பயிற்சி வினாக்கள்

1. தசம எண்ணை இருநிலை எண்ணாக மாற்ற ஒரு வழியைக் கூறு.

2. இருநிலை எண்ணைத் தசம எண்ணாக மாற்ற ஒரு வழி கூறு.

3. 11.081 என்பது இரு நிலை எண்ணா? எண்ணிலை எண்ணா? உன் விடைக்கான காரணம் என்ன?

4. கீழே உள்ள இருநிலை எண்களைத் தசம எண்களாக மாற்று.

   1011.0011   100.001   10001   110.011
   0.010101   1.1

5. கீழே உள்ள தசம எண்களை இருநிலை எண்களாக மாற்று.

   53.48   117   22.41   7.04   8.88   2.33

1.6 எண்ணிலை எண்களும் பதினாறு நிலை எண்களும்

எண்ணிலை எண்கள்

ஓர் எண்ணை இருநிலை எண்ணாக எழுதும் போது நிறைய இலக்கங்கள் தேவைப்படும். இதை நாம் படிப்பதும் நினைவில் நிறுத்துவதும் கடினம். இதைச் சற்று சுலபமாக்க உதவுவன எண்ணிலை எண்களும் பதினாறுநிலை எண்களும். இருநிலை எண்களிலிருந்து இவற்றுக்குப் போவதும் வருவதும் மிகச்சுலபம்.

இரு நிலை எண்ணில், புள்ளியில் தொடங்கி, இருபுறமும், மூன்று மூன்று இலக்கங்களாகச் சேர். முழு எண்ணின் இடதுபுறமும், பின்னத்தில் வலதுபுறமும் தேவையானால் சில சுழிகளையும் சேர்த்துக் கொள்ளலாம். ஒவ்வொரு மூன்று இலக்கத் தொகுதியையும் ஓர் இருநிலை எண்ணாகக் கருதினால், அது 0 முதல் 7 வரை ஓர் எண்ணாக இருக்கும். ஒவ்வொரு தொகுதியையும் இப்படி மாற்றி 0 முதல் 7 வரை உள்ள குறியீடுகளில் எழுதினால் அது எண்ணிலை எண்ணாக மாறிவிடும். வேறெதுவும் செய்யத் தேவையில்லை. உதாரணமாக, 1001110101.11001001111 என்பதை 001 001 110 101.110 010 011 110 என்று பிரித்து, அதை 1155.6235_8 என்ற எண்ணிலை எண்ணாக எழுதலாம். ஓர் எண்ணிலை எண்ணை இருநிலை எண்ணாக மாற்றுவதும் சுலபம். ஒவ்வொரு எண்ணிலை இலக்கத்தையும் சரியாக மூன்று இருநிலை இலக்கங்களாக எழுதினால் போதும். நன்றாகக் கவனியுங்கள் ஒவ்வொரு இலக்கமும் சரியாக மூன்று இலக்கங்களாக மாற வேண்டும். 1 என்பது 001 என்றும், 2 என்பது 010 என்றும் மாற வேண்டும். (கவனக் குறைவாக 2 என்பதை 10 என்று எழுதும் தவற்றை மாணவர்கள் அடிக்கடி செய்வார்கள்.)

பதினாறுநிலை எண்கள்

மூன்று இருநிலை இலக்கங்களைச் சேர்த்தால் ஓர் எண்ணிலை இலக்கம் கிடைக்கிறது. அதேபோல் நான்கு இருநிலை இலக்கங்களைச் சேர்த்தால் ஒரு பதினாறு நிலை இலக்கம் கிடைக்கும். அதாவது 0 முதல் 15 வரை உள்ள ஓர் எண் கிடைக்கும். இவற்றைக் குறிக்க மொத்தம் 16 குறியீடுகள் தேவை. நமக்குத் தெரிந்த குறியீடுகள் மொத்தம் பத்து, 0 முதல் 9 வரை. அதனால் புதிதாக 6 குறியீடுகள் தேவை. இதற்கு A, B, C, D, E, F என்ற குறியீடுகளைப் பயன்படுத்திக் கொள்வோம். ஆக, 0, 1, 2,...,9, A, B, C, D, E, F என்று 16 குறியீடுகள். இவற்றில் A = 10, B = 11, C = 12, D = 13, E = 14, F = 15, என்று மதிப்புகள் பெறும். உதாரணமாக 2B.D3 என்ற பதினாறு நிலை எண் $(11 \times 16^0) + (2 \times 16^1) + (12 \times 16^{-1}) + (3 \times 16^{-2})$ என்ற தசம எண்ணைக் குறிக்கும். எண்ணிலை, பதினாறு நிலை எண்களையும்கூட இடம் சார்ந்த குறியீட்டில்தான் எழுதுகிறோம்.

இருநிலை எண்களை பதினாறு நிலைக்கு மாற்றுவதும் சுலபம்தான். எண்ணிலைக்கு மாற்ற மூன்று இலக்கங்களாகத் தொகுத்ததுபோல் இங்கு நான்கு நான்காகத் தொகுக்க வேண்டும். வேறெந்த வித்தியாசமும் இல்லை. பதினாறுநிலை எண்ணை இருநிலைக்கு மாற்ற, ஒவ்வொரு இலக்கத்தையும் **சரியாக நான்கு** இருநிலை இலக்கங்களாக மாற்ற வேண்டும்.

எளிய வழி

ஓர் எண்ணை எந்த முறையிலிருந்தும், வேறொரு முறைக்கு மாற்றமுடியும். இதற்கு ஓர் எளிய வழி இருக்கிறது. முதலில் இருநிலை எண்ணாக மாற்றி, அதிலிருந்து தேவையான முறைக்கு மாற்றுவது.

தம எண் ↔ இருநிலை ↔ எண்ணிலை எண்

↕

பதினாறுநிலை எண்.

நினைவில் நிற்க

1. இருநிலை எண்ணை எண்ணிலை எண்ணாக மாற்ற, ஒவ்வொரு மூன்று இலக்கங்களையும் ஓர் எண்ணிலை இலக்கமாக மாற்ற வேண்டும். இப்படி மூன்று மூன்றாகத் தொகுக்கும்போது இருநிலைப்புள்ளிக்கு அருகிலிருந்து தொடங்க வேண்டும். தேவையானால் முழு எண்ணின் இடதுபுறமும், பின்னத்தின் வலது புறமும் சில சுழிகளைச் சேர்த்துக் கொள்ளலாம்.

2. நான்கு இருநிலை இலக்கங்கள் சேர்ந்து ஒரு பதினாறுநிலை இலக்கத்தைக் கொடுக்கும்.

பயிற்சி வினாக்கள்

1. கீழே வரும் எண்கள் சரியாக எழுதப்பட்டுள்ளனவா என்று கூறு

427.8_{10}	$1A\,6.7_{10}$	$AB.C_{16}$	$AM.B_{16}$	102.101_{2}
101.110_{2}	107.71_{8}	782.22_{8}	44.39_{8}	22.77_{8}

2. கீழே வரும் இருநிலை எண்களை எண்ணிலை, பதினாறு நிலை எண்களாக மாற்று

1100111.10111	1111.1111	110.1110011
10001.100111	10011000.0101	11110.110001

3. கீழே வரும் பதினாறு நிலை எண்களை இருநிலை எண்களாகவும், அவற்றிலிருந்து எண்ணிலை எண்களாகவும் மாற்று.

ABC.AE3	24F.FF	4CD.07
AA3.9B	FOB.00D	1AC.63

4. கீழே வரும் எண்ணிலை எண்களை பதினாறு நிலைக்கு மாற்று.

7301.511   654.321   123.765   101.101   267.741

1.7. எண்கணிதச் செயல்பாடுகள்

எண்களைப் புதிய குறியீடுகளில் எழுதினால் மட்டும் போதாது. அவற்றைக் கூட்ட வேண்டும், பெருக்க வேண்டும்; மேலும் கழிக்கவும், வகுக்கவும் வேண்டும். இருநிலை எண்களால் இது முடியுமா என்ற கேள்வி எழுகிறது. உண்மையில், தசம எண்களுக்குப் பதிலாக இருநிலை எண்களைக் கொண்டு கூட்டுவது, பெருக்குவது போன்றவை மிக எளிய செயல்கள். தசம எண்களுக்கு எத்தனை கூட்டல், பெருக்கல் வாய்ப்பாடுகளை மனப்பாடம் செய்திருக்கறோம்! இருநிலை எண்களுக்கு இந்த வாய்பாடுகள்தான் எவ்வளவு எளிமையானவை!



''முருகா, நாலாம் வாய்ப்பாடு மனப்பாடம் பண்ணியாச்சா?''
''இல்லை ஐயா, ஆனா இருநிலை எண் வாய்ப்பாடு எல்லாமே மனப்பாடம் பண்ணிட்டேன்''

0 + 0 = 0	0 X 0 = 0
0 + 1 = 1	0 X 0 = 0
1 + 0 = 1	1 X 0 = 0
1 + 1 = 10	1 X 1 = 1
இருநிலைக் கூட்டல் வாய்பாடு	**பெருக்கல் வாய்பாடு**

இங்கு 10 (பத்து என்று படிக்கக்கூடாது. ஒன்று, சுழி என்று படிக்க வேண்டும்.) என்பது 2 என்பது நினைவிருக்கட்டும். $1 + 1 = 10_2$. அதாவது, 1 + 1 என்று வரும் போது, அந்த இடத்தின் கூட்டுத் தொகை (Sum) 0 எனவும், அடுத்த இடத்திற்கு எடுத்துச் செல்ல வேண்டியது (Carry) 1 எனவும் ஆகிறது.

கூட்டல்

சில கூட்டல்களைப் பார்ப்போம். இங்கு, கூட்டப்படும் எண்களுக்கு மேலே சிறிய எழுத்துகளில் உள்ளது, அடுத்த இடத்திற்கு எடுத்துச் செல்லப்படும் எண்.

```
  11        1        1        11
  101      110       11       1010
+ 111     + 11     + 10     + 1111
------    ------   ------   ------
 1100      1001      101     11001
------    ------   ------   ------
5 + 7 = 12   6 + 3 = 9   3 + 2 = 5   10 + 15 = 25
```

பெருக்கல்

இருநிலைப் பெருக்கல்தான் எவ்வளவு சுலபம் பாருங்கள்.

```
     101          11         1010
  x  101       x 110       x 1001
  -------      ------      -------
     101          00         1010
    000          11         0000
   101          11         0000
  -------      ------      1010
   11001       10010      -------
  -------      ------     1011010
5 X 5 = 25   3 X 6 = 18   -------

                 10 X 9 = 90
```

தசம எண்களைப் போலவேதான் இங்கு கூட்டலும் பெருக்கலும். இச்செயல் பாடுகள் மிக எளிதாக இருந்தாலும், எண்களின் நீளம் அதிகம் என்பதுதான் ஒரு குறை. 90 என்பதற்கே 7 இலக்கங்கள் தேவைப்படுகின்றன.

கழித்தலும் வகுத்தலும்

இருநிலை முறையில் கடன் வாங்கிக் கழிக்கும்போது கவனிக்க வேண்டியது ஒன்று உண்டு. தசம முறையில் ஓர் இடத்திலிருந்து 1ஐக் கடன் வாங்கினால், அது அடுத்த (வலதுபக்க) இடத்தில் பத்தாக மாறும். ஆனால் இருநிலை முறையில் இது இரண்டாக மாறும். கீழே வரும் கழித்தல்களில் கடன் வாங்கிய எண் தனியாக சிறிய எழுத்துகளில் காண்பிக்கப்பட்டிருக்கிறது.

1	11	1	1111
101	100	1010	10000
- 10	- 11	-110	-1
011	001	0100	01111
5 - 2 = 3	4 - 3 = 1	10 - 6 = 4	16 - 1 = 15

வகுத்தலுக்குச் சில உதாரணங்கள்

4) 25 (6	3) 17 (5	100) 11001 (110	11) 10001 (0101
24	15	100	0
1	2	100	100
		100	11
		01	10
		0	0
		1	101
			11
			10

இருநிலை பின்னங்களையும் தசம பின்னங்களைப் போலவே பயன் படுத்தலாம்.

நினைவில் நிற்க

1. இருநிலை எண்களில் கூட்டல், கழித்தல், பெருக்கல், வகுத்தல் என்ற எண்கணிதச் செயல்பாடுகளை, தசம எண்களை வைத்துச் செய்வது போலவே செய்ய வேண்டும்.

2. இருநிலை எண்களில் கடன் வாங்கும்போது, ஒன்று என்பது அடுத்த இடத்தில் இரண்டாக மாறும்.

பயிற்சி வினாக்கள்

1. கீழே வருபவை இருநிலை எண்கள், தேவையான விடையைக் கண்டுபிடி.

1001 + 101	1001 - 11	101 x 11	1111 / 10
10011 + 1111	1110 - 111	111 x 1011	1011 / 10
11100 + 10011	1000 - 101	110 x 1001	11110 / 11
11 + 1011	1011 - 110	1000 x 1111	1001 / 11
1101 + 1000	1010 - 110	1011 x 1110	1101 / 111

1.8 எதிர்மறை எண்கள்

இப்போது கொஞ்சம் சிக்கலான பிரச்சனைக்கு வருவோம். இதுவரை நாம் பார்த்தவை எல்லாமே நேர்மறை எண்கள். எதிர்மறை எண்களை எவ்வாறு குறிப்பிடுவது? - 3 என்பதை இருநிலை எண்ணாக -11 என்று எழுதலாம். ஆனால் - (கழித்தல்குறி) என்ற ஒரு குறியீடு தேவையாகிறது. 0, 1 என்ற இரண்டு குறியீடுகளுடன், மூன்றாவது குறியீடு தேவையாகிறது. இலக்கவகைக் கணிப் பொறியில் இரண்டு குறியீடுகளுக்கு மேல் இல்லை. அதனால், கழித்தல் குறியைப் பயன்படுத்தாமலேயே எதிர்மறை எண்ணைக் குறிப்பிட முடியுமா என்று பார்ப்போம்.

எண்கள் முடிவற்றவை. ஆனால் நாம் பயன்படுத்தும் எண்கள் சாதாரணமாக ஒரு குறிப்பிட்ட எல்லைக்குள்தான் இருக்கும். (பெரிய பெரிய எண்கள் நமக்குத் தேவையா? அவற்றைப் பயன்படுத்துகிறோமா? - என்று ஒரு சுவையான, உபயோகமான பட்டிமன்றமே நடத்தலாம்.) இப்போதைக்கு, நாம் பயன்படுத்தும் பெரிய எண் 255 என்று வைத்துக் கொள்வோம். இதை இருநிலை எண்ணாகக் குறிக்க 8 இலக்கங்கள் தேவை ($255 = 1111\ 1111_2$). ஒவ்வொரு எண்ணையும் சரியாக ஒன்பது இலக்கங்களால் குறிப்பிடுவதாக முடிவு செய்வோம். இதில் இடதுபுற முதல் இலக்கம், அந்த எண் நேர்மறையா அல்லது எதிர்மறையா என்பதை உணர்த்தப் பயன்படும். மீதி 8 இலக்கங்கள் அந்த எண்ணைக் குறிக்கும். 0 முதல் 255 வரை உள்ள எல்லா நேர்மறை எண்களுக்கும், இடதுபுற முதல் இலக்கம் சுழியாக இருக்கட்டும், -1 முதல் -255 வரை உள்ள எதிர்மறை எண்களுக்கு இது 1 என்று இருக்கட்டும். இதன்படி

3 = 0 0000 0011	40 = 0 0010 1000
- 3 = 1 0000 0011	- 40 = 1 0010 1000

இங்கு இலக்கங்களுக்கு இடையில் இடம் விட்டிருப்பது நாம் எளிதாகப் படிப்பதற்காக. எல்லா எண்களுக்கும், சிறிய எண்ணாக இருந்தாலும் **சரியாக 9 இலக்கங்களைப்** பயன்படுத்தியாக வேண்டும். இது மிகமிக அவசியம்.

தப்புக்கணக்கு

ஒரு சின்னக் கணக்குப் போடலாம். 250 + 50 எவ்வளவு என்று பார்ப்போம்.

250 = 128 + 64 + 32 + 16 + 8 + 2

50 = 32 + 16 + 2

```
   0 1111 1010
 + 0 0011 0010
 -------------
   1 0010 1100
```

அதாவது 250 + 50 = -44 என்று வருகிறது. இது தவறல்லவா? இதற்குக் காரணம் என்ன? தொடக்கத்தில் நாம் எண்களைக் குறிப்பிட 9 இலக்கங்களைத் தேர்ந்தெடுத்தபோது, நாம் பயன்படுத்தும் மிகப்பெரிய எண் 255 என வைத்துக் கொண்டோம். இப்போது 300 என்ற எண்ணைக் குறிப்பிட முயல்கிறோம். அதனால் வந்த விளைவுதான் இது. இம்முறையில் எப்போதும் குறிப்பிட்ட எல்லைக்குள் உள்ள எண்களையே பயன்படுத்த வேண்டும்.

நினைவில் நிற்க

1. ஒரு குறிப்பிட்ட எல்லைக்குள் உள்ள எண்களைப்பற்றி மட்டும் பேசுவதாக இருந்தால், கூட ஓர் இலக்கத்தை இடதுபுறம் சேர்த்து, அதை நேர்மறையா அல்லது எதிர்மறையா என்று குறிப்பிடப் பயன்படுத்தலாம்.

2. இந்த முறையில் எல்லா எண்களும் ஒரே அளவு இலக்கங்களைப் பெற்றிருக்கும்.

பயிற்சி வினாக்கள்

1. கீழே வரும் எண்களை எத்தனை இலக்கங்கள் கொண்டு குறிப்பிட முடியும்? எப்படிக் குறிப்பிடுவது?

   அ. 4, 7, - 6, - 15, 12
   ஆ. 32, 46, - 62, 44, - 23
   இ. 100, 110, 6, - 120, - 45
   ஈ. 8, 30, - 27, 5

1.9 நிரப்பு முறைகள்

எதிர்மறை எண்களை, சென்ற பாடத்தில் பார்த்தபடி குறிப்பிடும்போதும் கூட்டல், பெருக்கல் முதலிய எண்கணிதச் செயல்பாடுகளைச் செய்யும்போதும் கவனத்துடன் இருக்கவேண்டும். இதற்கெனத் தனி வழிமுறைகளும் உண்டு. அவற்றை அப்படியே கணிப்பொறியில் செயல்படுத்துவது கடினம். அதனால் முழு எண்களைக் குறிப்பிடுவதில் மேலும் சில மாற்றங்கள் செய்யப்பட்டுள்ளன. இந்த மாற்றங்களினால், கணிப்பொறி எண்களைப் பயன்படுத்தும் வேகம் அதிகமாகிறது; கணிதச் செயல்பாடுகளின் விதிகள் எளிதாகின்றன. இந்தப் புது முறைகளைப்பற்றி ஓரளவு தெரிந்துகொள்வோம்.

ஒன்பதின் நிரப்பு முறை

முதலில் தசம எண்களைப் பார்ப்போம். மூன்று இலக்க எண்களைப்பற்றி மட்டும் பேசுவதாக வைத்துக்கொள்வோம். அதாவது அதிகபட்சம் + 999, குறைந்த பட்சம் - 999. நேர்மறை எண்ணுக்கு இடதுபுற முதல் இலக்கம் 0 என்று வைப்போம். இத்துடன் எண்ணைச் சரியாக மூன்று இலக்கங்களாக எழுதுவோம். உதாரணமாக் 27 என்பதை 0027 என்று எழுத வேண்டும். எதிர்மறை எண்ணை எழுதும்போது இரண்டு வேலைகள் செய்ய வேண்டும்.

1. இடதுபுற முதல் இலக்கம் 1 என எழுதவேண்டும்.

2. எதிர்மறை எண்ணின் **தனிமதிப்பை** (Absolute Value) அப்படியே எழுதாமல், அதை 999 இலிருந்து கழித்து எழுத வேண்டும். உதாரணமாக, - 26 என்பதை (999 - 26 = 973) 1973 என்றும், - 325 என்பதை 1674 என்றும் எழுதவேண்டும்.

எதிர்மறை எண்ணை, 999 வரை நிரப்ப எவ்வளவு சேர்க்க வேண்டும் என்பதைக் கணக்கிட்டு எழுதி வைப்பதாக நினைக்கலாம். அதனால் இந்த முறை **நிரப்பு முறை** (Complement notation) எனப்படும். 999 வரை நிரப்புவதால் இது ஒன்பதின் நிரப்பு முறை எனப்படும். இதை இன்னொரு விதமாகவும் நினைக்கலாம். அடுத்த பக்கத்திலுள்ள படத்தைப் பார்.

இந்த முறையை மொத்தமாகக் கடன் வாங்கி வைத்தல் என்றும் நினைக்கலாம். முதல் இலக்கம் கடன் வாங்கப்பட்டுள்ளதா என்பதைக் குறிக்கும்.

999 எவ்வாறு வந்தது? நாம் மூன்று இலக்க எண்களைப் பற்றிப் பேசுவதால், அத்தனை இலக்கங்களிலும் பெரிய எண்ணான ஒன்பதைப் போட்டிருக்கிறோம். இதுதான் 3 இலக்கப் பெரிய எண். அதற்குமேல் கடன் வாங்கத் தேவையில்லை! இதைப் போலவே 5 இலக்க எண்களைப் பற்றிப் பேசினால் 99999 என்று வரும்.



"உங்களிடம் எவ்வளவு பணம் இருக்கிறது?"
முதல்வர் : பணமா? 325 ரூபாய் கடன்தான் இருக்கிறது.
.இரண்டாமவர் : யாராவது 999 ரூபாய் கடன் கொடுத்தால், இருக்கும் கடன்போக, மீதி 674 ரூபாய் கையில் இருக்கும்.
மூன்றாமவர் : யாராவது 1,000 ரூபாய் கடன் கொடுத்தால், இருககும் கடன் போக, மீதி 675 ரூபாய் கையில் இருக்கும்.

பத்தின் நிரப்புமுறை

அது என்ன 999 ரூபாய் கடன் வாங்குவது? 1000 ரூபாய் என்றால் முழுமையாக இருக்ககுமே? - என்று நினைக்கிறார் மூன்றாமவர். அப்படியும் செய்யலாம். இம்முறையில் எழுதும்போது இடதுபுற முதல் இலக்கம் 1 என்றால் 1000 கடன் இருக்கிறது என்பதைக் குறிக்கும். எதிர்மறை எண்ணின் தனி மதிப்பை ஆயிரத்திலிருந்து கழித்து, அடுத்த மூன்று இலக்கங்களாக எழுத வேண்டும். இவ்விதத்தில்,

- 26 = 1974, - 325 = 1675 என்று வரும்.

நேர்மறை எண்களில் எந்த மாற்றமும் இருக்காது.

26 = 0026   325 = 0325 என்று வரும்.

பத்தின் நிரப்பு முறையில் ஓர் எதிர்மறை எண் = (ஒன்பதின் நிரப்பு முறையில் அந்த எண்) + 1 என்பது தெளிவாகத் தெரிகிறதா?

ஒன்றின், இரண்டின் நிரப்பு முறைகள்

இருநிலை எண்களில் பெரிய இலக்கம் 1. அதற்கடுத்த எண் (10_2 இரண்டு. அதனால் இருநிலை முறைகளில் **ஒன்றின் நிரப்பு முறையையும்** (One's Complement), **இரண்டின் நிரப்பு முறையையும்** (Two's Complement) பற்றிப் பார்க்க வேண்டும். நான்கு இலக்க எதிர்மறை எண்களைப்பற்றி பேசும் போது முறையே 1111 அல்லது 10000 கடன் வாங்க வேண்டும். இந்த முறைகளில். நேர்மறை எண்களில் எந்த மாற்றமும் இல்லை, சரியான அளவு இலக்கங்களைப் பயன்படுத்துவதைத் தவிர. சில உதாரணங்கள் -

பட்டியல் 1 - 4

தசம எண்	4 இலக்க இருநிலை எண்	ஒன்றின் நிரப்புமுறை	இரண்டின் நிரப்பு முறை
3	= 0011	= 00011	= 00011
-3	= - 0011	= 11100	= 11101
10	= 1010	= 01010	= 01010
- 10	= - 1010	= 10101	= 10110
- 15	= - 1111	= 10000	= 10001
0	= 0000	= 00000	= 00000
- 0	= - 0000	= 11111	= 00000
- 16	= - 10000	குறிப்பிடமுடியாது	= 10000

பட்டியலில் கடைசி மூன்று வரிசைகளைக் கவனித்துப் பாருங்கள். ஒன்றின் நிரப்புக் குறியீட்டில் 0, - 0 இரண்டுக்கும் வெவ்வேறு குறியீடுகள்; - 16ஐக் குறிப்பிட முடியாது. ஆனால் இரண்டின் நிரப்புக் குறியீட்டில் 0, - 0 இரண்டுக்கும் ஒரே குறியீடு; அத்துடன் - 16ஐயும் குறிப்பிட முடியும் (மொத்தம் 10000_2 = 16 கடன் வாங்குவதால்).

எளிய வழி

ஓர் இருநிலை எண்ணை ஒன்றின் நிரப்பு முறைக்கு மாற்ற ஓர் எளிய வழி உள்ளது. முதலில் +, - குறிகளை நீக்கிவிட்டு அந்த எண்ணுக்கு முன்னால் ஒரு 0 சேர். எண் நேர்மறை என்றால் வேறெதுவும் செய்ய வேண்டாம்; எண் எதிர்மறை என்றால், எல்லா இலக்கங்களையும், 0 என்றால் 1 என்றும், 1 என்றால் 0 என்றும் மாற்று.

ஓர் இருநிலை எண்ணை இரண்டின் நிரப்புமுறைக்கு மாற்ற, முதலில் +, - குறிகளை நீக்கிவிட்டு, இடதுபுறம் ஒரு 0 சேர். எண் நேர்மறை என்றால் வேறெதுவும் செய்ய வேண்டாம்; எண் எதிர்மறை என்றால், வலது புறத்திலிருந்து இடதுபுறம் நோக்கி வரவேண்டும்; அப்போது வரும் முதல் 1க்குப்பிறகு வரும் ஒவ்வொரு இலக்கத்தையும், 0 என்பதை 1 என்றும், 1 என்பதை 0 என்றும் மாற்று.

இரண்டின் நிரப்புமுறை, கணிப்பொறிகளில் பெரிதும் பயன்படுகிறது. இம்முறையில் கணிதச் செயல்பாடுகளுக்கு எளிய முறைகள் உள்ளன. இம்முறையில் எல்லா எண்களும் ஒரே நீளமாக இருப்பது கணிப்பொறியின் வேலையை எளிதாக்குகிறது.

நிரப்புக் குறியீடுகளின் வரையறைகளை இப்படிச் சுருக்கமாகக் கூறலாம். எண்களின் அதிகபட்ச இலக்கங்கள் n என இருக்கட்டும். x என்பது n இலக்க எண்.

x ஒரு தசம எண்

ஒன்பதின் நிரப்புக் குறியீட்டில், $x = \begin{cases} 0x, & x \text{ நேர்மறை என்றால்}, \\ 1(10^n - 1 - x), & x \text{ எதிர்மறை என்றால்}. \end{cases}$

பத்தின் நிரப்புக் குறியீட்டில், $x = \begin{cases} 0x, & x \text{ நேர்மறை என்றால்}, \\ 1(10^n - x), & x \text{ எதிர்மறை என்றால்}. \end{cases}$

x ஓர் இருநிலை எண்

ஒன்றின் நிரப்புக் குறியீட்டில், $x = \begin{cases} 0x, & x \text{ நேர்மறை என்றால்}, \\ 1(2^n - 1 - x), & x \text{ எதிர்மறை என்றால்}. \end{cases}$

இரண்டின் நிரப்புக் குறியீட்டில், $x = \begin{cases} 0x, & x \text{ நேர்மறை என்றால்} \\ 1(2^n - x), & x \text{ எதிர்மறை என்றால்}. \end{cases}$

கவனித்தீர்களா? எதிர்மறை எண்களுக்கு, ஒன்றின் நிரப்புக் குறியீட்டுடன் ஒன்றைச் சேர்த்தால் இரண்டின் நிரப்புக் குறியீடு கிடைக்கும்.

நினைவில் நிற்க

1. மொத்தமாகக் கடன் வாங்கி, உள்ள கடனை அடைத்துவிட்டு, மீதியைக் கூறுவது, நிரப்புமுறை எனப்படும்.

2. இரண்டின் நிரப்புமுறை கணிப்பொறியில் பெரிதும் பயன்படுகிறது.

3. இம்முறைகளில் எல்லா எண்களும் ஒரே நீளமாக இருக்கும்.

4. இம்முறைகளை முழு எண்களுக்கு மட்டும் பயன்படுத்துகிறோம்.

பயிற்சி வினாக்கள்

1. இருநிலை எண்ணை ஒன்றின் நிரப்பு முறைக்கு மாற்ற ஒரு வழி கூறு.

2. இருநிலை எண்ணை இரண்டின் நிரப்பு முறைக்கு மாற்ற ஒரு வழிகூறு.

3. கீழே வரும் தசம எண்களை ஒன்பதின் நிரப்பு முறையிலும் பத்தின் நிரப்புமுறையிலும் எழுது. மொத்த இலக்கங்கள் 3 + 1 எனக் கொள்.

236, 428, -7, -325, -900, 28, -28, 8.

4. கீழே வரும் இருநிலை எண்களை ஒன்றின் மற்றும் இரண்டின் நிரப்பு முறைகளில் எழுது. மொத்த இலக்கங்கள் 3+1 எனக் கொள்.

1001 0110	-0000 1000	-1111 0000	-0111 1100
110011	-1001 1101	-0101 0101	-1100 0111

5 கீழே வரும் ஒன்றின் நிரப்பு முறை எண்களை இருநிலை எண்களாகவும், தசம எண்களாகவும் மாற்று.

000101	100011	001	00111	111
111111	1111	1001	1000	100

6 கீழே வரும் இரண்டின் நிரப்புமுறை எண்களை இருநிலை எண்களாகவும் தசம எண்களாகவும் மாற்று.

100011	0011	10101	01011	0001
11111	1111	1001	1000	100

1.10 நிரப்புமுறைக் கணிதம்

எல்லா முழு எண்களையும், கழித்தல் குறி இல்லாமலேயே நிரப்பு முறையில் எழுத முடியும் என்று பார்த்தோம். இதனால் என்ன நன்மை? கணிப்பொறி கூட்டலையும், கழித்தலையும் விரைவாகச் செய்ய முடியும். எப்படி? முதலில் ஓர் ஆச்சரியமான விஷயத்தைத் தெரிந்துகொள்வோம். மிச்சமுறைக் கணிதத்தில் கூட்டல் மட்டுமே உண்டு. கழித்தலே இல்லை! அப்படியானால் 5 - 3 என்பது போன்றவற்றை எப்படிச் செய்வது? 5 - 3 என்பதை 5 + (- 3) என்ற கூட்டலாக! மொத்தம் 5 இலக்கங்களைப் பயன்படுத்தும் ஒன்றின் நிரப்புமுறையில் முதலில் செய்து பார்ப்போம்.

5 + (- 3) = 2	6 + 4 = 10	3 + (- 5) = - 2	(-3) + (- 5) = - 8
0 0101	0 0110	0 0011	1 1100
+ 1 1100	+ 0 0100	+ 1 1010	+ 1 1010
10 0001	0 1010	11101	11 0110
↳ + 1			↳ 1
00010			10111

இந்த முறையில் நாம் பயன்படுத்தும் 5 இலக்கங்களுக்கும் மேலாக ஓர் இலக்கம் வந்தால், அதை விடையுடன் கூட்ட வேண்டும்.

நினைவிருக்கிறதா? நிரப்புமுறைகளில் எல்லா எண்களும் ஒரே நீளமிருக்கும். குறிப்பிட்ட எல்லைக்குள் உள்ள எண்களைத்தான் கையாள முடியும். இரு எண்களைக் கூட்டும்போது இந்த எல்லையைவிட்டு வெளியே போனால் என்னவாகும்? ஏற்கனவே ஒரு முறை பார்த்தது போல தவறான விடைதான் வரும். உதாரணமாக, மொத்தம் 5 இலக்கங்களைப் பயன்படுத்தினால் ஒன்றின் நிரப்பு முறையில் -15 முதல் + 15 வரைதான் கையாளலாம். 9 + 8 = 17 அல்லது - 10 - 6 = - 16 என்ற கணக்குகளைச் செய்யமுடியாது. செய்ய முயன்றால் தவறான விடைகள்தான் வரும்.

9 + 8 = 17

  0 1001

+ 0 1000

  1 0001 = - 4 (தவறான விடை)

(- 10) + (- 6) = - 16

```
  1 0101
+ 1 1001
--------
1 0 1110
└→     1
--------
  0 1111   = 15 (தவறான விடை)
```

கூட்டும்போது ஓர் இலக்கம் அதிகமாக வந்தால், அதை மறுபடி கூட்ட வேண்டும் என்பது ஒன்றின் நிரப்பு முறையில் ஒரு பெரும் குறை. இரண்டின் நிரப்பு முறையில் இந்தக் குறை இல்லை. கூட்டும்போது கூட ஓர் இலக்கம் வந்தால், கவலைப்படாமல் அதைத் தூக்கி எறிந்து விடலாம்! மறுபடி கூட்ட வேண்டியதில்லை. சில உதாரணங்கள்.

```
5 + (- 3) = 2           6 + 4 = 10    3 + (- 5) = -2    (-3) + (-13) = -16
  0 0101                  0 0110        0 0011            1 1101
+ 1 1101                + 0 0100      + 1 1011          + 1 0011
----------              --------      --------          ----------
10 0010 (விடை)          0 1010        1 1110            1 1 0000 (விடை)
↑                                                       ↑
விட்டு விடு                                              விட்டு விடு
```

வரும் விடைகள் - 16 முதல் + 15 வரை இருந்தால் சரியான விடையாக இருக்கும். இல்லையென்றால் தவறான விடை வரும்.

உதாரணமாக,

```
  9 + 8 = 17          - 10 - 7 = - 17
  0 1001                 1 0110
+ 0 1000                 1 1001
--------              ---------
  1 0001  = -15        1 0 1111  = 15   தவறான விடை
                       ↑
தவறான விடை             விட்டு விடு
```

இந்த நிரப்பு முறைகளின் சிறப்பான அம்சம் இதுதான். நேர்மறை, எதிர் மறைடயக் குறிக்கும் குறியீடுகளை, சாதாரண 0, 1களாக எழுதுவதோடு மட்டுமல்லாமல், அவற்றை சாதாரண இருநிலை எண்களைப் போலவே கூட்டவும் செய்கிறோம்! அவற்றைத் தனியாகப் பிரித்துப் பார்க்காததால் கணிப்பது எளிதாகிறது.

பெருக்கல்

இரண்டின் நிரப்புமுறையில் பெருக்குவதும் எளிது. அதற்கு ஒரு வழியை இங்கு பார்ப்போம். எண்களை வகுக்கும்போது இரண்டையும் நேர்மறை எண்களாக எடுத்துக்கொள்ள வேண்டும். தேவையானால் நேர்மறை/எதிர்மறைக் குறிகளை விடையுடன் சேர்த்துக் கொள்ள வேண்டும். அதனால் வகுத்தல் பற்றி இங்கு பார்க்கப் போவதில்லை.

பெருக்கும்போது கீழ்க்காணுமாறு செய்ய வேண்டும்.

1. இரண்டும் நேர்மறை - சாதாரணப் பெருக்கல் போல் செய்.

2. இரண்டும் எதிர்மறை - இரண்டையும் நேர்மறையாக்கி, சாதாரணப் பெருக்கலாகச் செய்.

3. ஒன்று நேர்மறை, ஒன்று எதிர்மறை - **முதல் எண்ணை எதிர்மறையாகவும், இரண்டாவதை நேர்மறையாகவும்** எடுத்துக்கொள். இரண்டு n இலக்க எண்களைப் பெருக்கினால் 2n இலக்க எண் வரும். ஓர் எண்ணை, ஓர் இலக்கத்தால் பெருக்கி எழுதும்போது, அந்த எண்ணின் இடதுபுறத்தில் உள்ள இடங்களை, அந்த எண்ணின் இடதுபுற முதல் இலக்கத்தால் நிரப்பு. மற்றபடி சாதாரணப் பெருக்கலைப்போல் செய். (நேர்மறை எண்களுக்கு இடதுபுறத்தில் தேவையான அளவு 0 உம், எதிர்மறை எண்களுக்கு இடது புறம் 1உம் சேர்க்க வேண்டும்).

உதாரணமாக,

-3 X 2 = -6	-4 X 3 = -12
1 01	100
X 0 10	X 011
0000 00	111100
1110 1	11100
0000	000
1110 10	1110100

நினைவிருக்கட்டும். இங்கு முதல் எண்தான் எதிர்மறையாக இருக்க வேண்டும். இல்லையென்றால்? தவறான விடை வரும் என்பதைச் சோதித்துப்பார்.

நினைவில் நிற்க

1. நிரப்பு முறைகளில் கூட்டல் உண்டு, கழித்தல் இல்லை.

2. ஒன்றின் நிரப்பு முறைக் கூட்டலில், கூட ஓர் இலக்கம் வந்தால், அதைத் திரும்பவும் கூட்ட வேண்டும். இரண்டின் நிரப்பு முறையில் அதை விட்டுவிட வேண்டும்.

3. இரண்டின் நிரப்புமுறைப் பெருக்கலில் ஓர் எண் மட்டும் எதிர்மறையாக இருந்தால், அது பெருக்கலில் முதல் எண்ணாக இருக்க வேண்டும்.

4. இரண்டின் நிரப்புமுறை கணிப்பொறிகளில் பயன்படுகிறது.

பயிற்சி வினாக்கள்

1. கீழ்க்காணும் கணக்குகளுக்கு ஒன்றின் நிரப்பு முறையிலும், இரண்டின் நிரப்பு முறையிலும் விடைகாண்.

    (அ) 5 இலக்கங்களைப் பயன்படுத்து

    5 + 6, 6 - 8, - 9 + 4, 14 - 6, 15 - 11, 10 + 9, - 9 - 6

    (ஆ) 4 இலக்கங்களைப் பயன்படுத்து

    4 + 2, - 2 + 3, - 4 - 3, - 2 + 6, - 2 - 5

    (இ) 4 இலக்கங்களைப் பயன்படுத்தி இரண்டின் நிரப்பு முறையில் செய்

    - 7 x 4, 7 x 5, - 5 x 4, 6 x - 3, 4 x 5

1.11 மிதக்கும் புள்ளி முறை

பிட்டும், பைட்டும்

இருநிலை இலக்கம் என்பது ஆங்கிலத்தில் binary digit என்று வரும். இதைக் குறுக்கி *(பிட்)* (bit) என்பார்கள். எட்டு பிட்டுகள் சேர்ந்து வருவதை ஒரு பைட் (byte) என்பார்கள். சாதாரணமாக, பல பிட்டுகள் சேர்ந்து வரும்போது அவற்றை பைட்டுகளின் மடங்காகக் கூறுவார்கள். உதாரணமாக 2 பைட், 4 பைட்.

பெரிய எண்கள்

சென்ற பாடங்களில் முழு எண்களை எப்படி 0, 1 கொண்டு குறிப்பிடுவது என்று பார்த்தோம். கணிப்பொறியில், சாதாரணமாக ஓர் எண்ணைக் குறிப்பிட மொத்தம் 16 பிட்டுகளை, அதாவது 2 பைட்டுகளைப் பயன்படுத்துகிறார்கள். இதில் நேர்மறை/எதிர்மறையைக் குறிக்க ஒரு பிட் போய்விடும். மீதி 15 பிட்டுகளைக் கொண்டு அதிகபட்சம் $2^{15} - 1 = 32,767$ என்ற எண்ணைக் குறிக்கலாம். இரண்டின் மிச்ச முறையில் - 32,768 முதல் + 32,767 வரை குறிப்பிடலாம். இந்த எல்லைக்கு வெளியே உள்ள பெரிய எண்களைக் குறிப்பிட வேண்டுமானால்,இன்னும் நிறைய,



"உங்கள மாசாதது எவ்வளவு?"
முதல்வர் : நான்கு கோடியே இருபத்து மூன்று ரூபாய் ஐம்பது காசுகள்.
இரண்டாமவர் : நான்கு கோடி ரூபாய்.

நூற்றுக்கணக்கில், பிட்டுகள் வேண்டியிருக்கும். இதற்கு இடம் அதிகமாவதுடன், கணக்கிடும் நேரமும் அதிகமாகும். அதனால் பெரிய எண்களைக் குறிக்க இந்த முறை உகந்ததில்லை. ஒரு புதிய முறை வேண்டும்.

கோடிக்கணக்கில்

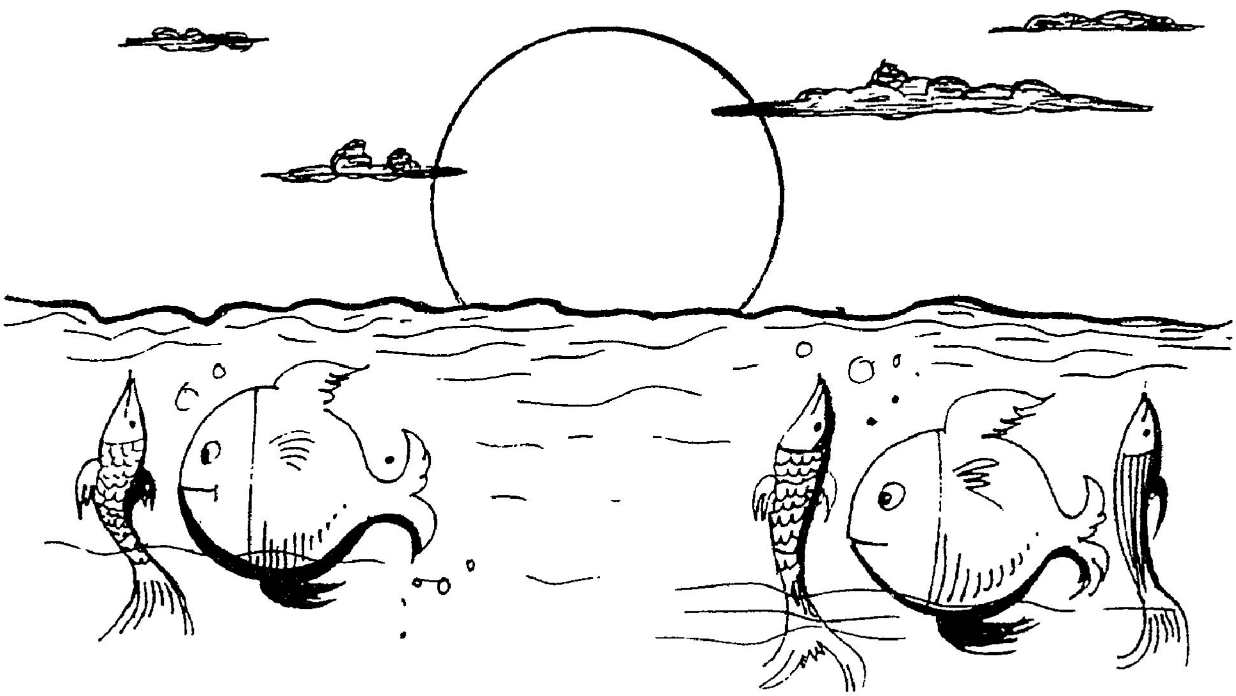
கோடிக்கணக்கில் பேசும்போது, சில நூறுகள் மதிப்பிழந்து காணப்படும். பல லட்சம் கோடிகள் பற்றிப் பேசும்போது, ஒன்றிரண்டு கோடிகளுக்கு மதிப்பில்லை. ஓர் எண்ணுக்கு மதிப்புள்ளதா இல்லையா என்பது, பேசப்படும் இடத்தைப் பொறுத்து உள்ளது. இங்கு நாம் அந்த எண்ணின் உண்மையான மதிப்பைக் குறிப்பிடவில்லை. கொடுத்த சூழ்நிலையில் ஒப்பு நோக்கி வரும் மதிப்பைக் குறிப்பிடுகிறோம். அதாவது, இந்தச் சூழ்நிலையில் சில இலக்கங்களைச் சொல்லாமல் விட்டுவிடுவதனால் ஏற்படும் இழப்பு குறிப்பிடும்படியாக இல்லாவிட்டால், அந்த இலக்கங்களுக்கு மதிப்பில்லை, விட்டுவிடலாம் எனக்கூறலாம். தலைப்பில் 'லட்சங்களில்' அல்லது 'கோடிகளில்' என்று எழுதுவார்கள். இங்கு கோடிகளில் குறிப்பிடும்போது 1,025 என்பது 1,025 கோடிகளைக் குறிக்கும். அதாவது 1025,00,00,000. இங்கு சில இலக்கங்களை எழுதாமலேயே விட்டு விடுகிறோம். இதனால் தேவைப்படும் இடம் குறைகிறது.

கணிப்பொறி சாதாரணமாக ஓர் எண்ணின் இடதுபுற, அதாவது அதிக மதிப்புள்ள, 7 **தசம** இடங்களை மட்டும் நினைவில் வைத்துக்கொள்ளும். உதாரணமாக 123456789 என்ற எண்ணை, 123456800 என்று ஏழு இலக்கங்களுக்குச் சரி செய்யும். 789 என்பது 800 என்று முழுமையாக்கப்படும். பிறகு இது 1234568×10^2 என்றும், 0.1234569×10^9 என்றும் மாற்றப்படும். இதை 0.1234568 E9 என்று எழுதுவார்கள். இதிலிருந்து 1234568 என்பதையும் 9 என்பதையும் கணிப்பொறி நினைவில் வைக்கும். இங்கு எப்போதும் 'பத்தின் அடுக்கு' (Power of 10) என்று வருவதால் 10 என்பதை நினைவில் வைக்கத் தேவையில்லை. அதேபோல் தசமப்புள்ளியானது எண்ணின் இடதுபுறம் வருவதால் அதையும் நினைவில் வைக்கத் தேவையில்லை.

எதிர்மறை எண்ணான -1234×10^{45} என்பதை - 0.1234E49 என்று எழுத வேண்டும். பெரிய எண்களை எழுத ஏற்படுத்திய இந்த முறை மிகச்சிறிய எண்களுக்கும் பயன்படும்.

0.000...(மொத்தம் 30 சுழிகள்)...123 என்ற எண்ணை

0.123×10^{-30} என்றும், அதை 0.123E -30 என்றும் எழுதலாம்.



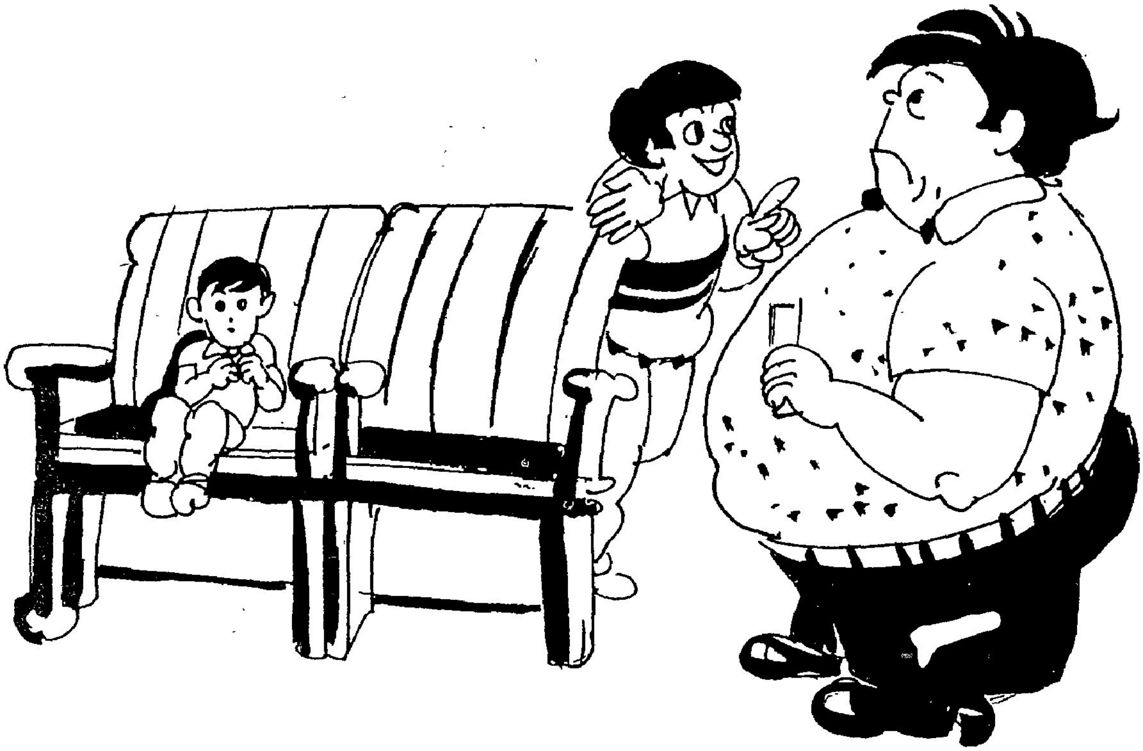
"எவ்வளவு பெரிய மிதக்கும் புள்ளி!"

மிதக்கும் புள்ளி

இந்த குறியீட்டில் தசமப் புள்ளியானது எப்போதும், அதிக மதிப்புள்ள, சுழியில்லாத முதல் இலக்கத்திற்கு இடதுபுறம் சென்று நிற்கும். தசமப் புள்ளி இப்படி இடம் நகர்ந்து செல்வதால் இதை **மிதக்கும் புள்ளிமுறை** (floating point notation) என்பார்கள். பத்தின் அடுக்கைக் குறிப்பிடுவதால் **அடுக்குக் குறியீடு** (Exponential notation) என்றும் இது பெயர்பெறும்.

வாமனன் முதல் விஸ்வரூபம் வரை

*மிதக்கும் புள்ளி முறையில் 32 பிட்டுகளைக் கொண்டு எவ்வளவு பெரிய, எவ்வளவு சிறிய எண்களைக் குறிப்பிடலாம் என்று பார்ப்போம். எண்ணின் +, - குறிக்கு ஒரு பிட். 7 **தசம** இலக்கங்களுக்கு 24 பிட்டுகள். அடுக்கின் +, - குறிக்கு ஒரு பிட். மீதி 6 பிட்டுகள் அடுக்கினைக் குறிக்க. இந்த 6 பிட்டுகளினால் 0 முதல் $2^6 -1 = 63$ வரை குறிப்பிடலாம். அதனால் இந்த முறையில் குறிப்பிடக்கூடிய பெரிய எண் 0.9999999×10^{63}. மிகச்சிறிய எண் 0.1×10^{-63}. மிதக்கும் புள்ளி முறையில், 32 பிட்டுகளில் வாமனனுக்கும் இடம் உண்டு. விஸ்வரூபத்திற்கும் இடம் உண்டு.*



"ஒரு டிக்கட்டுக்கு ஒரு இடந்தான் கொடுக்க முடியும். இதிலே உட்காருங்க."

என்ன இழந்தோம்?

ஆச்சரியப்படத்தக்க வகையில் இவ்வளவு நன்மைகளைக் கொடுத்த இந்த முறையில் உள்ள பாதக அம்சம் என்னவென்றால், குறிப்பிடும் எண்ணில் ஒரு மிகச்சிறிய பகுதியை இழக்கலாம் என்பதுதான். இப்படி இழப்பது குறிப்பிடும் படியாக இருக்காது என்று பார்த்தோம். ஆனால் சிறுதுளி பெருவெள்ளம் என்பது போல், சின்னச் சின்ன இழப்புகள் மிகப்பல சேர்ந்தால், அது பேரிழப்பில் முடியும் என்பதை நன்றாகக் கவனத்தில் வைத்துச் செயல்பட வேண்டும். இல்லாவிட்டால் கணக்கிற்கு நாம் எதிர்பார்க்கும் விடை இல்லாமல் வேறெதாவது வந்து நிற்கும். ஓர் உதாரணத்தால் இதை விளக்குவோம்.

கணிப்பொறி 7 தசம இலக்கங்களை மட்டும் நினைவில் வைத்துக் கொள்வதாக இருக்கட்டும். A = 76,54,321, B = 0.01 என்று இருக்கட்டும். A + 10,000 B என்பதன் மதிப்பென்ன?

$10,000 \times 0.01 = 100$

$76,54,321 + 100 = 76,54,421.$

இதையே A + B + B.... (பத்தாயிரம் முறை) என்றும் எழுதலாம். கணிப்பொறி ஒரு சமயத்தில் ஒரு செயலைச் செய்யும். முதலில் A உடன் ஒரு Bஐக்கூட்டும். பிறகு ஒரு Bஐ. இப்படி மொத்தம் பத்தாயிரம் முறைகள் செய்யும்.

A + B = *76,54,321 + 0.01*
= *0.7654321*E7 *+ 0.1* E*-1*
= *0.7654321* E7 *+ 0.000000001* E7 *= 0.765432101* E7
= *0.7654321* E7 *(ஏழு இலக்கங்களுக்குச் சரியாக)*

அதாவது A + B *என்பது* A*க்குச் சமம் என்று விடை வருகிறது! இப்படி ஒவ்வொரு முறையும் ஒரு* B*ஐ இழந்து விடுவோம். கடைசியில் பத்தாயிரம்* B*க்களையும் இழந்து,*

A + B + B + ... = A *என்று வரும் !*

விழிப்புடன் இரு

மேலே உள்ள உதாரணத்தை வைத்து "இந்த மிதக்கும் புள்ளி முறையைப் பயன்படுத்தி கணிப்பொறி எப்போதும் தப்பான விடையைக் கொடுக்கும் போலிருக்கிறதே?" என்று எண்ண வேண்டாம். இப்படிப்பட்ட இடங்கள் வரும்போது, அதாவது மிகப்பெரிய எண்ணுடன் ஒரு மிகச் சிறிய எண்ணைக் கூட்டும்போது, விழிப்புடன் இருக்க வேண்டும். பல சிறு எண்களைக் கூட்டிய பிறகு அதை பெரிய எண்ணுடன் கூட்ட வேண்டும். இப்படிச் செய்தால் மேலே பார்த்த தவறுகளைத் தவிர்த்து விடலாம்.

விசித்திரம்

வழக்கமாக நாம் பின்ன எண்களைத் தவிர்த்து முழு எண்களாக வைத்துக் கொள்ள முயல்வோம். ஆனால் கணிப்பொறியானது மிகப்பெரிய எண்களைப் பின்னமாக மாற்றி வைத்துக்கொள்கிறது!

நினைவில் நிற்க

1. மிகப் பெரிய எண்ணையும், மிகச் சிறிய எண்ணையும் குறிப்பிட மிதக்கும் புள்ளி முறை (அடுக்குக் குறியீடு) உதவுகிறது.

2. கணிப்பொறியில் மிகப்பெரிய எண்களுடன் மிகச் சிறிய எண்களைப் பயன்படுத்தும்போது கவனத்துடன் இருக்கவேண்டும்.

பயிற்சி வினாக்கள்

1. கீழ் வரும் தசம எண்களை மிதக்கும் புள்ளி முறையில் எழுது.

73	1673456789012	3.00000002	0 - 0.123
0.000007	0.0010000008	101.102	

0.00....(26 சுழிகள்) 8899001122334

2. கீழே வரும் எண்களைச் சாதாரணத் தசம எண்களாக எழுது.

- 0.79 E - 6	0.81234E - 3	0.1234 E 3
0.11 E- 5	- 0.1E 4	0.2345 E 8
0.14 E13	- 0.999 E 9	0.121 E 5

3. 100 (A + B) என்பதற்கும் 100 A + 100 B என்பதற்கும் கணிப்பொறி ஒரே விடையைத்தருமா? எப்போது வெவ்வேறு விடைகளைத் தரும்? எப்படிக் கணக்கிடுவது நல்லது?

4. 4 பைட்டுகளைக் கொண்டு மிதக்கும் புள்ளி முறையில் எண்களைக் குறிப்பிடுவது எவ்வாறு? இம்முறையில் எவ்வளவு பெரிய, எவ்வளவு சிறிய எண்களைக் குறிப்பிடலாம்?

1.12 ஆஸ்கி குறியீடுகள்

0,1 என்ற குறியீடுகளைக் கொண்டே, எல்லா எண்களையும், +, - குறி உட்பட, குறிப்பிடலாம் என்று பார்த்தோம். கணிப்பொறியின் திரையில் எண்களுடன், ஆங்கில எழுத்துக்களையும், *, < போன்ற சிறப்புக் குறியீடுகளையும் கூடப் பார்க்கிறோம். கணிப்பொறி இவைகளை அச்சடித்தும் கொடுக்கிறது. இவற்றை யெல்லாம் கூட 0, 1 க்குள் அடக்க முடியுமா என்ற கேள்வி எழுகிறது. ஆம். அடக்க முடியும்!

ஆஸ்கி குறியீடு

8 பிட்டுகளில் மொத்தம் 256 எண்களை எழுதலாம்; 0 முதல் 255 வரை. இந்த ஒவ்வொரு எண்ணுக்கும் ஒரு குறியீட்டை ஈடாகக் கூறினால், மொத்தம் 256 குறியீடுகளைக் குறிப்பிடலாம். பெரும்பாலும் நமக்கு இதற்கு மேல் தேவையிருக்காது. 0 முதல் 127 வரை உள்ள எண்களுக்கு ஏற்படுத்தப்பட்ட ஒரு குறியீடு இன்று உலகம் முழுவதும் பயன்படுகிறது. இது ASCII (ஆஸ்கி) குறியீடு எனப்படும். American Standard Code for Information Interchange என்பதன் சுருக்கம் இது. இதை பட்டியல் 1.5 இல் காணலாம்.

இந்த ஆஸ்கி குறியீட்டைக் கொண்டு, ஆங்கில வார்த்தைகள் அத்தனையையும் எழுதலாம். உதாரணமாக, GET என்பதை 71 69 84 என்று குறிப்பிடலாம். A < C என்பதை 65 60 67 என்று குறிப்பிடலாம். 0 முதல் 9 வரை உள்ள தசம இலக்கங்களுக்கும் கூட ஆஸ்கி குறியீடுகள் உள்ளன என்பதைக் கவனிக்கவும். எண்களை உள்ளீடு, வெளியீடு செய்யும்போது இது பயன்படும். எண்கணிதச் செயல்பாடுகளுக்கு இந்தக் குறியீடு உகந்ததில்லை. அதற்கு எண்களை நாம் ஏற்கனவே பார்த்த முறைகளில் மாற்றிக்கொள்ள வேண்டும். அல்லது ஆஸ்கி குறியீடுகளை வைத்தே கூட்ட, பெருக்க புதிய வழிமுறைகளைக் கையாள வேண்டும். ஆஸ்கி குறியீட்டில் 237 என்பதை 50 51 55 என்று குறிப்பிட வேண்டும்.

இஸ்கி குறியீடுகள்

ஆஸ்கி குறியீட்டில் 0 முதல் 127 வரை போக, மீதி உள்ள 128 முதல் 255 வரை உள்ள எண்களுக்கு, இந்திய மொழிகளில் உள்ள எழுத்துக்களை ஈடாக வைத்து, ஆஸ்கி குறியீட்டை விரிவுபடுத்தியிருக்கிறார்கள். இதற்கு ISCII (இஸ்கி - Indian Standard Code for Information Interchange) என்று பெயர். இந்தக் குறியீடு இந்திய மொழிகள் எல்லாவற்றிற்கும் பொருந்தும்.

ஆஸ்கி குறியீட்டைப் போன்ற, EBCDIC (Extended Binary Coded Decimal Interchange Code) எப்ஸிடிக் குறியீடு IBM 360, 370 கணிப்பொறிகளில் பயன்படுத்தப்பட்டது.

பட்டியல் 1-5

DEC	HEX	CHR	DEC	HEX	CHR	DEC	HEX	CHR	DEC	HEX	CHR
0	0	none	32	20	-	64	40	@	96	60	
1	1	none	33	21	/	65	41	A	97	61	a
2	2	none	34	22	!!	66	42	B	98	61	b
3	3	none	35	23	#	67	43	C	99	63	c
4	4	none	36	24	$	68	44	D	100	64	d
5	5	none	37	25	%	69	45	E	101	65	e
6	6	none	38	26	&	70	46	F	102	66	f
7	7	bell	39	27	'	71	47	G	103	67	g
8	8	BS	40	28	(	72	48	H	104	68	h
9	9	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	-	78	4E	N	110	6E	n
15	0F	SL	47	2F	/	79	4F	O	111	6F	o
16	10	none	48	30	0	80	50	p	112	70	p
17	11	none	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	none	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	none	53	45	5	85	55	U	117	75	u
22	16	none	54	36	6	86	56	V	118	76	v
23	17	none	55	37	7	87	57	w	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	none	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	:	91	5B	[	123	7B	{
28	1C	none	60	3C	<	92	5C	\	124	7C	|
29	1D	none	61	3D	=	93	5D	]	125	7D	}
30	1E	none	62	3E	>	94	5E	∧	126	7E	~
31	1F	none	63	3F	?	95	5F	-	127	7F	DEL

ஆஸ்கி அட்டவணையில் தசம எண்களுடன் பதினாறுநிலை எண்களும் குறிப்பிடப்பட்டிருப்பதைக் கவனித்திருப்பீர்கள். எட்டு பிட்டுகள் இரண்டு நான்கு இலக்கங்களாகப் பிரிக்கப்பட்டு, இரண்டு பதினாறுநிலை எண்களாகக் கொடுக்கப்பட்டிருக்கின்றன. படிப்பதற்கு எளிதாக இருப்பதற்காக இப்படிக் கொடுக்கப்படுகிறது. இவற்றை நேரடியாகத் தசம எண்களாக மாற்றலாம், இடம் சார்ந்த குறியீட்டு முறையை நினைவில் கொண்டு. உதாரணமாக,

$$2A = (2 \times 16^1) + 10 = 42.$$

நினைவில் நிற்க

1. ஆங்கில எழுத்துகள், தசம எண்கள் மற்றும் சிறப்புக் குறியீடுகளை 0, 1 மூலம் குறிப்பிட ஆஸ்கி குறியீடு பயன்படுகிறது.

2. இந்திய மொழி எழுத்துக்களைக் குறிப்பிட இஸ்கி குறியீடு உருவாக்கப்பட்டுள்ளது.

3. ஆஸ்கி குறியீட்டில் எண்கள் இருந்தால், கணிதச் செயல்பாடுகளைச் செய்வது சற்றுக் கடினம்.

பயிற்சி வினாக்கள்

1. A, B, a, b, 0, 1, 2, 5, +, -, *, என்பவற்றின் ஆஸ்கி குறியீடுகளை எழுது.

2. 6237 என்ற எண்ணை ஆஸ்கி குறியீட்டில் எழுது.

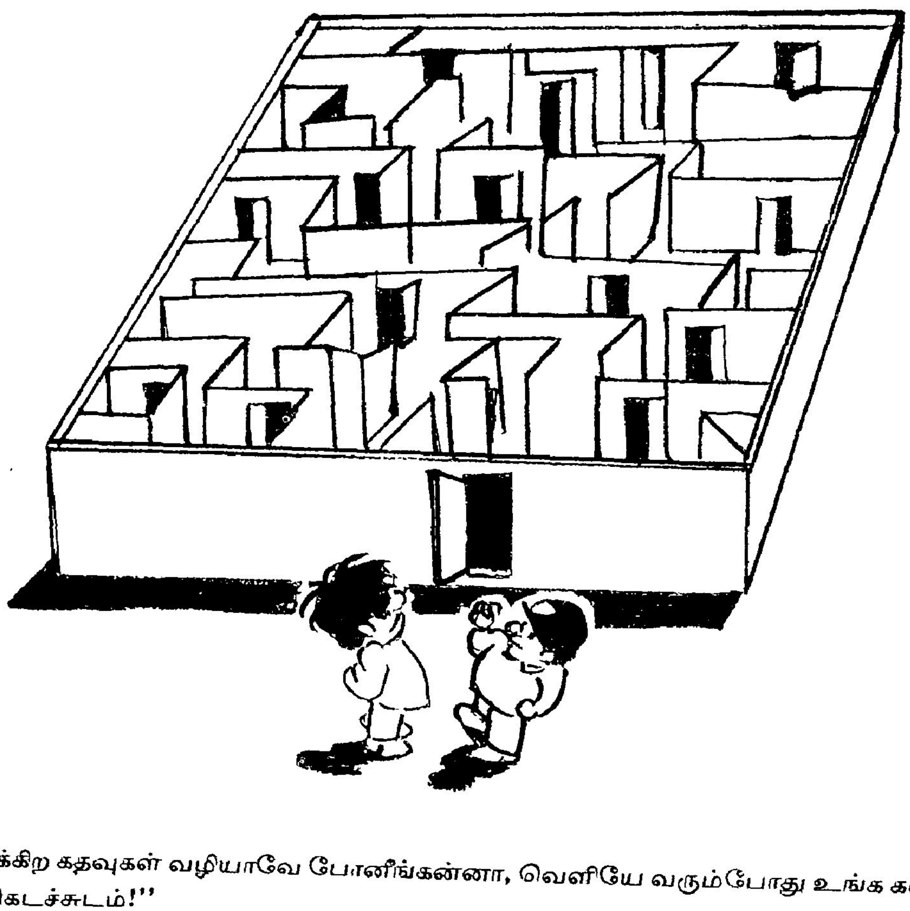
3. 48, 65, 97 என்ற ஆஸ்கி குறியீடுகள் எவற்றைக் குறிக்கின்றன?

1.13 பூலியன் கணிதம்

இலக்கவகைக் கணிப்பொறியை இயக்கும் அடிப்படைக் கணித அமைப்பைப் பற்றி இந்தப் பாடத்தில் பார்ப்போம்.

சுவிட்சுகள் (Switches)

இன்றைய கணிப்பொறியின் அடிப்படை அமைப்புகளில் மிக முக்கியமான ஒன்றைக் கூறவேண்டுமானால், அது நமக்கு ஏற்கனவே தெரிந்த சுவிட்சுதான். ஆம். கணிப்பொறியின் உள்ளே நடப்பதெல்லாம் பல்லாயிரக்கணக்கான சுவிட்சுகளை திறந்து மூடுவதுதான். ஒரு வயலுக்கு நீர் பாய்ச்ச, பல மடைகளைத் திறந்தும், மீதியை மூடிவைத்தும், நீரைச் சரியான இடத்திற்குத் திருப்பிவிடுகிறோம். இதைப் போலவே பல சுவிட்சுகளை மூடியும், பலவற்றைத் திறந்தும் வைக்கும்போது, பாயும் மின்சாரத்தின் பாதையும் மாறுகிறது. கடைசியில் பாயும் இடங்கள் நம் விடையைக் குறிக்கின்றன.



''திறந்திருக்கிற கதவுகள் வழியாவே போனீங்கன்னா, வெளியே வரும்போது உங்க கணக்குக்கு விடை கெடச்சுடம்!''

பல்லாயிரக்கணக்கான சுவிட்சுகள் - ஒவ்வொன்றையும் ஒரு வினாடியில் கோடிக்கணக்கான முறைகள் திறந்து மூடுவது - இதுதான் இன்றைய கணிப்பொறி! தவறாக எழுதவில்லை. இன்றைக்கு சில சில்லுகளில் இந்தக் கதவுகளை திறந்து மூட ஆகும் நேரம் நானோ அல்லது பீக்கோ வினாடிகளில் இருக்கிறது.

நானோ வினாடி = $1/10^9$ வினாடி

பீக்கோ வினாடி = $1/10^{12}$ வினாடி

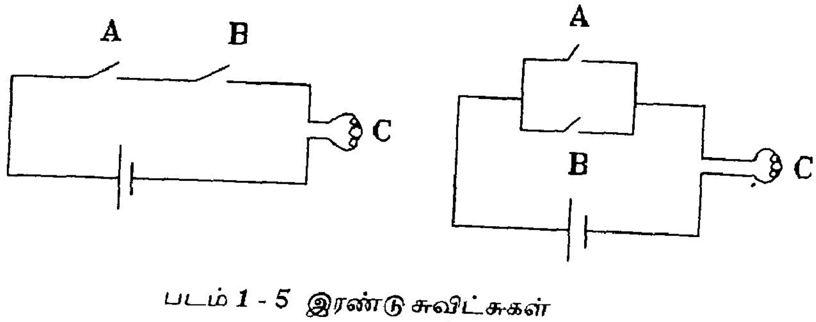
இந்த சுவிட்சுகள் எல்லாமே நகரும் வகையில்லை. மின்னணு வகை. ஒரு கம்பியில் மின்சாரத்தைச் செலுத்தினால் அது ஒரு சுவிட்சைப் போடும், அல்லது நிறுத்தும். இந்த சுவிட்சுகளுக்கான கணிதம்தான் கணிப்பொறியை இயக்கும் கணிதம்; மிகவும் எளிதான ஒன்று. பூலியன் கணிதம் (Boolean Algebra) என்று பெயர். பூல் (George Boole) என்பவரால் 19 ஆம் நூற்றாண்டில் கண்டுபிடிக்கப் பட்டது. தர்க்கவியலுக்கும் பயன்படுகிறது.

இந்தக் கணிதத்தின் அடிப்படையும் 0, 1 என்ற இரண்டு குறியீடுகள்தான். இவற்றை தசம அல்லது இருநிலை இலக்கங்களுடன் குழப்பிக் கொள்ளாதீர்கள். இவை புதிய குறியீடுகள்.

0 என்பது ஒரு சுவிட்ச் திறந்திருக்கிறது (மின்சாரம் பாயவில்லை) என்பதையும்

1 என்பது சுவிட்ச் மூடியிருக்கிறது (மின்சாரம் பாயலாம்) என்பதையும் குறிக்கும்.

சாதாரணமாக ஒரு விளக்கிற்கு ஒரு சுவிட்சுதான் இருக்கும். நாம் ஒரு விளக்கிற்கு இரண்டு சுவிட்சுகளைப் பொருத்திப் பார்ப்போம். இதை படம் 1.5 இல் உள்ளதுபோல் இரு வழிகளில் செய்யலாம். முதல் படத்தில் சுவிட்சுகள் ஒன்றுக்கடுத்து ஒன்றாக உள்ளன (Series Connection). இரண்டாவதில் இரண்டும் இணையாக உள்ளன (Parallel connection). A, B என்ற சுவிட்சுகள் இருக்கும் நிலையைப் பொறுத்து C என்ற விளக்கு எரியும். இதைப் பட்டியலிட்டுக் காட்டலாம் (பட்டியல் 1.6). முதல் படத்தில் இரண்டு சுவிட்சுகளும் மூடியிருந்தால் மட்டும் விளக்கெரியும். இரண்டாம் படத்தில் A, அல்லது B, அல்லது இரண்டுமே மூடியிருந்தால் விளக்கு எரியும். A, B என்ற இரண்டு சுவிட்சுகளைக் கட்டுப்படுத்துவதன் மூலம், C என்ற இடத்தில் மின்சாரம் பாய்வதைக் கட்டுப்படுத்துகிறோம்.



படம் 1 - 5 இரண்டு சுவிட்சுகள்

பட்டியல் 1.6

A	B	C = A and B	A	B	C = A or B
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	1	1	1

இதை இருவிதங்களில் செய்கிறோம். இந்த அமைப்புகளை **அண்ட் கதவு** (AND gate) மற்றும் **ஆர் கதவு** (OR gate) என்பார்கள் இந்தப் பட்டியல்களைப் பார்த்தால், இவை வாய்ப்பாடுகள் போலிருக்கின்றன அல்லவா? இரண்டு இருநிலை எண்களைச் சேர்த்து ஒரு புதிய இருநிலை எண்ணை உருவாக்குவது போல்தானே இருக்கிறது? இந்தப் பட்டியல்கள், அதாவது வாய்ப்பாடுகள் இரண்டு எண்கணிதச் செயல்பாடுகளைக் குறிப்பதாக நினைப்போம். இந்தச் செயல்பாடுகளை AND, OR என்போம். AND என்பதை, சுருக்கமாக, பெருக்கல் குறியாலும், OR என்பதை சுருக்கமாக கூட்டல் குறியாலும் குறிப்பிடுவோம். மேலே பார்த்த பட்டியல்களை இப்போது வாய்ப்பாடுகளாகவே எழுதி விடலாம். இப்படி -

0.0=0 *0+0=0*

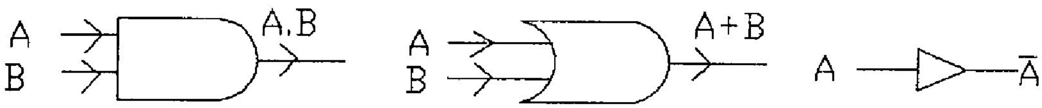
0.1=0 *0+1=1*

1.0=0 *1+0=1*

1.1=1 *1+1=1*

இதில் ஒரு வரியைத் தவிர மற்றவையெல்லாம், சாதாரண பெருக்கல், கூட்டல் போல் இருக்கின்றன அல்லவா? இந்த இரண்டு கதவுகளையும், மின்னணுப் பகுதிகளாக வடிவமைக்க முடியும். அவற்றை படம் 1.6 இல் உள்ளதுபோல் குறிப்பிடுவார்கள். இத்துடன் சுழியை ஒன்று என்றும், ஒன்றை சுழி என்றும் மாற்றும் இன்வெர்ட்டர் (Inverter) என்ற கதவும் உள்ளது. இந்தச் செயல்பாடு Complementing எனப்படும். A என்பதை A பார் (Bar) அல்லது A காம்ப்ளிமெண்ட் (Complement) என்று படிக்க வேண்டும். இங்கு ஒரு எண்ணிலிருந்தே இன்னொரு எண் உருவாவதால் இது ஓர்உறுப்புச் செயல்பாடு (Unary Operation). AND மற்றும் OR ஈருறுப்புச் செயல்பாடுகள் (Binary Operation) இந்த மூன்று செயல்பாடுகளும் சேர்ந்து உருவாக்கும் கணிதம்தான் பூலியன் கணிதம் எனப்படும்.

இந்த மூன்று வகைக் கதவுகளையும், அவற்றின் சேர்க்கைகளையும், மாற்றங்களையும் மட்டும் வைத்தே, எல்லாக் கணக்குகளையும் செய்யவல்ல கணிப்பொறி கட்டப்பட்டிருக்கிறது! ஆச்சரியமாக இருக்கிறதா? எப்படிக் கட்டப்பட்டிருக்கிறது என்பதைப் படிப்படியாகப் படித்தால் புரியும். அடுத்த பகுதியில் இதைப் பார்ப்போம்.



படம் *1 - 6* கதவுகள்

நினைவில் நிற்க

1. இரண்டு சுவிட்சுகளை வைத்ததுபோல், மின்சாரம் செல்வதைக் கட்டுப்படுத்தும் சிறு சுற்றுகளுக்கு, கதவுகள் என்று பெயர்.

2. அடிப்படைக் கதவுகள் மூன்று - அண்ட், ஆர், காம்ப்ளிமென்ட்.

3. இந்தக் கதவுகளைத் தேவையானபடி சேர்ப்பதால் கணிப்பொறி உருவாகிறது.

பயிற்சி வினாக்கள்

1. ஆர் கதவு செயல்படும் விதத்தை விளக்கு. அதன் செயல்பாட்டை ஒரு பட்டியலாகக் கூறு.

2. அண்ட் கதவு என்றால் என்ன? அதன் செயல்பாட்டை ஒரு சுற்றின் மூலமும், ஒரு பட்டியல் மூலமும் விவரி.

3. காம்ப்ளிமெண்ட் கதவு என்ன செய்யும்?

4. ஓர்உறுப்புச் செயல்பாட்டிற்கும் ஈருறுப்புச் செயல்பாட்டிற்கும் உள்ள வித்தியாசம் என்ன? உதாரணத்துடன் விளக்குக.

கணிப்பொறியின் பாகங்கள்

2

கணிப்பொறியின் அமைப்பைக் கூறுபடுத்திப் பார்க்கும்போது, அதில் மையச் செயலகம், நினைவகம், உள்ளீட்டகம், வெளியீட்டகம் என்று நான்கு பெரும் பகுதிகள் தெரியும். மையச் செயலகத்தைக் கட்டுப்பாட்டகம், கணித தர்க்கச் செயலகம், பதிவிடங்கள் என்று மேலும் பகுக்கலாம். உள்ளீட்டகங்கள், வெளியீட்டகங்கள் மற்றும் வெளிநினைவகங்களில் பல வகைகள் உண்டு. இப்பகுதியில் இவற்றைப் பற்றித் தெரிந்துகொள்வதுடன், கணிப்பொறித் துறையில் பயன்படும் பொது வார்த்தைகள் பலவற்றைப் பற்றியும் அறிந்து கொள்வோம். தொகுப்புமுறை, நேரடி முறை, நேரப்பகிர்வு முறை, பலபணி முறை, உடனடி முறை, நிகழ்காலச் செயல்பாட்டு முறை, PC, XT, AT, மோடம், பஸ், பின்னலமைப்பு, இயக்க முறை போன்றவற்றைப் பற்றியும் சிறிது தெரிந்துகொள்வோம்.

2.1 கணிப்பொறியின் அமைப்பு

கணிப்பொறியின் அமைப்பில் 5 முக்கிய பகுதிகள் உள்ளன. அவை **கணித தர்க்கச் செயலகம்** (Arithmetic Logic Unit), **கட்டுப்பாட்டகம்** (Control Unit), **நினைவகம்** (Memory), **உள்ளீட்டகம்** (Input Unit), **வெளியீட்டகம்** (Output Unit). *இவற்றில் கணித தர்க்கச் செயலகம், கட்டுப்பாட்டகம் என்பவற்றுடன் சில பதிவிடங்களைச் சேர்த்து, அதை* **மையச் செயலகம்** (CPU - Central Processing Unit) *என்றும் கூறலாம். நாம் ஏற்கனவே பார்த்த எண் கணிதம், பூலியன் கணிதம் மற்றும் மூன்று வகைக் கதவுகளையும் கொண்டு, கணித தர்க்கச் செயலகத்தை எப்படி உருவாக்குவது என்பனீத முதலில் பார்ப்போம். பின்பு மற்றப் பகுதிகளின் தேவையையும், செயல்பாட்டையும் பார்ப்போம்.*

கூட்டல் சுற்று

எண்கணிதச் செயல்பாடுகளில் முதன்மையான கூட்டலை எடுத்துக் கொள்வோம். இரண்டு இருநிலை இலக்கங்களைக் கூட்டும்போது, அந்த இடத்திற்கு வரும் கூட்டுத் தொகையையும் (sum), *அடுத்த இடத்திற்கு எடுத்துச் செல்லும் எண்ணையும்* (carry) *எப்படி உருவாக்குவது என்று பார்ப்போம். இந்த இரண்டையும் இரண்டு சார்புகளாக எழுதலாம்.*

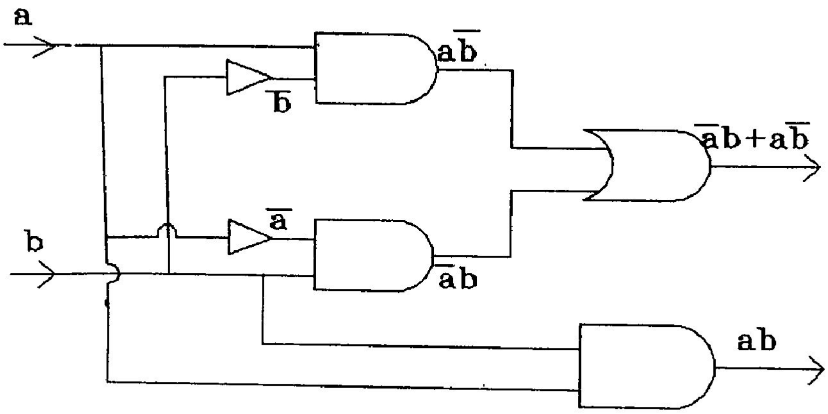
a	b	carry	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

இதைப் பார்த்த உடனேயே தெரிகிறது carry = a.b *என்று. இங்கு . (புள்ளி) என்பது பூலியன் செயல்பாடான அண்ட்.* sum *என்பது இரண்டு, மூன்றாம் வரிசைகளில் மட்டும் 1 என்ற மதிப்பினைப் பெறுகிறது. அதனால் அதை*

sum = $\bar{a}.b + a.\bar{b}$ *என்று எழுதலாம்.*

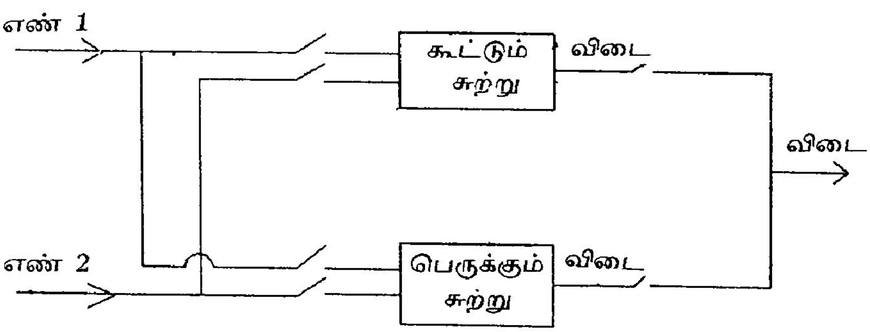
a = 0, b = 1 *என்றால்* sum = 1 *என்ற வரியிலிருந்து* $\bar{a}\,b$ *கிடைக்கிறது. இதேபோல் அடுத்தவரியிலிருந்து* $a.\bar{b}$ *கிடைக்கிறது. இந்த* $\bar{a}.b + a.b$, a.b *என்ற பூலியன் கோவைகளை* (Booban Expressions) *படம் 2.1 இல் உள்ளதுபோல் ஒரு மின்சுற்றாக மாற்றலாம்.*

கூட்டும்போது, இரண்டு இலக்கங்களுடன், முந்தைய இடத்திலிருந்து வந்த இலக்கத்தையும் சேர்த்துக்கொள்ள வேண்டும். இந்த மூன்றையும் வைத்துத்தான் கூட்டுத்தொகையையும், அடுத்த இடத்திற்குச் செல்லும் இலக்கத்தையும் கணக்கிட வேண்டும். இதற்கான சுற்றையும் மேலே பார்த்தது போலவே உருவாக்கலாம்.



படம் 2 - 1 கூட்டல் சுற்று

இம்மாதிரிச் சுற்றுகளைத் திரும்பத்திரும்பப் பயன்படுத்தி, பல இலக்க எண்களைக் கூட்ட ஒரு சுற்றை அமைக்கலாம். இதேபோல் பெருக்கல், வகுத்தல், போன்றவற்றிற்கும் தனித்தனியாகச் சுற்றுகள் அமைக்கலாம். எந்தச் செயல்பாடு தேவையோ அந்தச் சுற்றுக்கான வழியை மட்டும் திறந்து விட்டு மற்றவற்றை மூடி வைத்தால், மின்சாரம் அந்தச் சுற்று வழியாக மட்டும் சென்று தேவையான விடையைக் கொடுக்கும். படம் 2.2 இதை விளக்குகிறது.



படம் 2 - 2 கணிக்கும் பகுதி

கணித தர்க்கச் செயலகம்

முழு எண்களைக் கூட்டவும் பெருக்கவும் சுற்றுகளைச் செய்ததுபோலவே, பின்னங்களுக்கும் கூட சுற்றுகளை உருவாக்கலாம். இதோடு, இரு எண்கள் சமமா அல்லது எது பெரியது என்று கண்டுபிடிக்கவும் சுற்றுகளை ஏற்படுத்தலாம். இவை

எல்லாவற்றையும் ஒரே இடத்தில் வைத்து, தேவையான செயல்பாட்டைத் தேர்ந்தெடுத்து அதைச் செய்து கொள்ளலாம். இந்தச் சுற்றுகளின் தொகுதி **கணித தர்க்கச் செயலகம்** (ALU - Arithmetic Logic Unit) எனப்படும்.

ALU இல்லாமல் கணிப்பொறி இல்லை. இது ஒரு நல்ல பணியாள். இதற்கு வேலை கொடுக்க, அதாவது எந்தத் தகவலை என்ன செய்ய வேண்டும் என்று சொல்ல வேறு யாராவது வேண்டும். இப்பணியைச் செய்வது **கட்டுப்பாட்டகம்** (Control Unit). பெயருக்கேற்றாற்போல் இப்பகுதி மற்ற எல்லாப் பகுதிகளையும் கட்டுப்படுத்தி வேலை வாங்கும். கட்டுப்பாட்டகம், கணித தர்க்கப் பகுதிக்கு தகவல்களைக் கொடுத்து வாங்கும்போது, அவ்வப்போது தகவல்களை சிறிது நேரத்திற்கு வைத்திருக்க சில நினைவிடங்கள் தேவை. இவை **பதிவிடங்கள்** (Registers) எனப்படும். சாதாரணமாக 8 அல்லது 16 பதிவிடங்கள் ஒரு கணிப்பொறியில் இருக்கும். இத்தனை போதுமா? சில நூறுகளாவது வேண்டாமா? - என்று கேட்கத் தோன்றுகிறதா? நல்ல கேள்விதான். சில பதிவிடங்கள் போதுமானதே. ஆனால், இப்போது வர ஆரம்பித்திருக்கும் RISC (Reduced Instruction Set Computer) கணிப்பொறிகளில் அதிக அளவு பதிவிடங்களைப் பயன்படுத்தி கணிப்பதை விரைவுபடுத்தி வருகிறார்கள். கட்டுப்பாட்டகம், கணித தர்க்கச் செயலகம், பதிவிடங்கள் இவை மூன்றும் சேர்ந்து, **மையச் செயலகம்** (CPU - Central Processing Unit) எனப்படும்.

நினைவகம் (Memory Unit)

கணிப்பொறி செயல்பட பலவிதமான தகவல்கள் அதன் நினைவில் இருக்க வேண்டும். கணக்குகளில் வரும் எண்கள், குறியீடுகள், சரங்கள் முதலியன, இந்தக் கணக்குகளைச் செய்யும் வழிமுறைகள், கணிப்பொறியின் மேலாளராகப் பணிபுரியும் **இயக்க அமைப்பு** (Operating System) - இவை எல்லாமே நினைவில் இருக்க வேண்டும். கணிப்பொறிக்கு இவை எல்லாமே 0, 1 கொண்ட சரங்கள்தான். இவற்றை 8 பிட்டுகள் சேர்ந்த பைட்டுகளின் சரங்களாகப் பார்க்கலாம்.

கணிப்பொறியின் நினைவகத்தில் பல அறைகள் இருக்கும். ஒவ்வொன்றிலும் ஒரு பைட்டை வைக்கலாம். ஒரு தகவலை எங்கு வைக்கிறோம் என்பது தெரிந்தால்தானே அதைத் திரும்பவும் எடுக்க முடியும்? அதனால் நம் வீடுகளுக்கு எண்கள் கொடுப்பதுபோல், இந்த அறைகளுக்கும் எண்கள் உண்டு. இந்த அறை எண்கள் சுழியில் ஆரம்பித்து 1,2... என்று செல்லும். இன்றைய மேஜைக் கணிப்பொறிகளில் 640 KB நினைவகம் இருப்பது மிகச் சாதாரணம். இங்கு K என்பது கிலோ என்பதன் சுருக்கம். சாதாரணமாக கிலோ என்பது ஆயிரத்தைக் குறிக்கும். ஆனால் கணிப்பொறி இயலில் 1K என்பது $1024 = 2^{10}$ ஐக் குறிக்கும். கிட்டத்தட்ட என்று கணக்கிடும்போது ஆயிரம் என்றே எடுத்துக்கொள்ளலாம்.

இதேபோல் 1M என்பது (1 மெகா-Mega) என்பது 10^{20}, அதாவது 1024 x 1024, கிட்டத்தட்ட ஆயிரம் ஆயிரங்கள் (1,000,000). இங்கு B என்பது பைட்டுகளைக் குறிக்கிறது. மேஜைக் கணிப்பொறிகள் இன்று சுமார் 16 MB நினைவகங்களுடன் கிடைக்கின்றன.

சொல் நீளம்

ஒவ்வொரு பைட்டாக எடுத்து வேலை செய்தால் மிகவும் தாமதமாகும். இதைத் தவிர்க்க பெரும்பாலும் 4 பைட்டுகளை ஒரே சமயத்தில் நினைவகத்திற்கு அனுப்புவார்கள், அல்லது அங்கிருந்து எடுத்துக்கொள்வார்கள். ஒரே சமயத்தில் 32 பிட்டுகள் இணையாகப் போய்வர 32 இணையான கம்பிகள் தேவை. இப்படி இணையாகச் செல்லும் கம்பிகள் **பஸ்** (Bus) எனப்படும். பஸ்களின் அகலம் அதிகமாக ஆக, கணிக்கும் வேகமும் அதிகமாகும். ஆனால், இத்தனைக் கம்பிகளையும் சில்லுக்குள்ளேயும், **தாய்ப்பலகையிலும்** (Mother Board) அடைப்பது மிகவும் கடினமான செயலாகிக்கொண்டே போகும்.

கணிப்பொறி ஒரு சமயத்தில் 4 பைட்டுகளை வைத்துக் கணக்கிட்டால், அக்கணிப்பொறி பயன்படுத்தும் சொல்லின் (வார்த்தையின்) நீளம், அதாவது கணிப்பொறியின் **சொல்நீளம்** (Word length) 4 எனப்படும். இப்போதைக்கு சொல் நீளம் 4 என வைத்துக் கொள்வோம். முதல் வார்த்தையை (சொல்லை), அதாவது 4 பைட்டுகளை, அறை எண் 0 இலிருந்து தொடங்கி, 0,1,2,3 என்ற எண்ணுள்ள அறைகளில் வைக்கலாம். அடுத்த வார்த்தைகள் 4,8,12,16,... என்ற எண்ணுள்ள அறைகளில் தொடங்கும். எந்த வார்த்தையும் நாலின் மடங்காக உள்ள எண் அறையில்தான் தொடங்கும். இந்தத் தொடக்க அறை எண்ணே 4 அறைகளைக் குறிப்பதாக எடுத்துக்கொள்ளலாம்.

ஒரு வார்த்தையை நினைவகத்தில் வைக்க, அந்த வார்த்தையையும், அதன் முகவரியையும் (4 இன் மடங்கான ஒரு எண்) கொடுக்க வேண்டும். அதேபோல் ஒரு வார்த்தையைப் பெற, ஒரு முகவரியைக் கொடுத்து இந்த அறையில் ஆரம்பிக்கும் வார்த்தையை எடுத்துக்கொடு என்று கேட்கலாம்.

இப்படி ஆணையிட்டதும், அதைச் செய்து முடிக்கும்படி உள்ள ஒரு சுற்றின் தொகுதிதான் இந்த **நினைவகம்** (Memory) என்னும் பகுதி. ஆம். இதுவும் ஒரு சுற்றுதான் (Circuit). ஒவ்வொரு அறையும் 8 சிறு சுற்றுகளின் தொகுதி. ஒவ்வொரு சுற்றிலும் ஒரு பிட்டை எழுதி நிறுத்தி வைக்கலாம், அழித்து எழுதலாம்; என்ன எழுதியிருக்கிறதென்று பார்க்கலாம்.

தகவல்களை, குறிப்பிட்ட இடத்தில் வைத்து, கேட்கும்போது எடுத்துக் கொடுப்பதுதான் நினைவகத்தின் பணி. நினைவகத்திற்கு வரும் தகவல் ஒரு உள்ளீட்டகத்திலிருந்தோ அல்லது மையச் செயலகத்திலிருந்தோ வரலாம். நினைவகத்திலிருந்து வெளியே வரும் தகவல் மையச் செயலகத்திற்கோ அல்லது

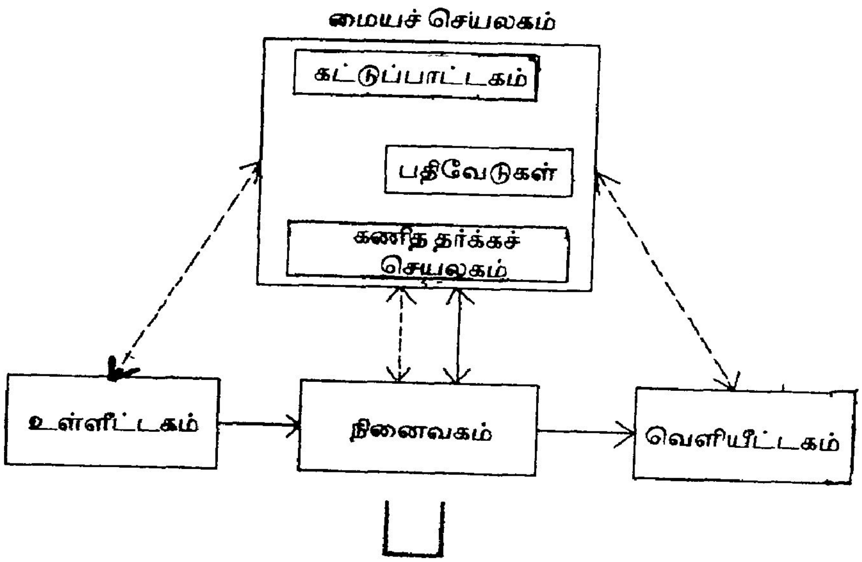
ஒரு வெளியீட்டகத்திற்கோ செல்லலாம். இப்படி, தகவல் எங்கிருந்து எங்கே செல்ல வேண்டும் என்பதை, பல சுவிட்சுகளைத் திறந்து மூடிக் கட்டுப்படுத்துவது கட்டுப்பாட்டகத்தின் வேலை.

உள், வெளியீட்டகங்கள் (Input Output Units)

நம் கணக்குகளுக்குத் தேவையான தகவல்களையும், முறைகளையும் கணிப் பொறிக்கு உள்ளே செலுத்த வேண்டும். இதற்குப் பலவித சாதனங்கள் உள்ளன. *சாவிப்பலகை* (Key board), *நெகிழ்தகடு இயக்கி* (Floppy disk drive), *துளை அட்டைப் படிப்பான்* (Card Reader), *மின்னணுப் பலகை* (Electronic Tablet) போன்றவை இதில் அடங்கும்.

வெளியீட்டகம் இல்லாமல் கணிப்பொறி முற்றுப் பெறாது. *திரை* (VDV - Video Display Unit), *தகடு இயக்கிகள்* (Disk Drives), *காந்த நாடா இயக்கிகள்* (Tape Drives), *அட்டை துளைப்பான்கள்* (Card Punching Machines), *பலவித அச்சுப் பொறிகள்* (Printers), *வரைவான்கள்* (Plotters) போன்றவை இதில் அடங்கும்.

இதுவரை பார்த்த 4 பகுதிகளும் கணிப்பொறியின் அடிப்படைப் பகுதி களாகும். படம் 2-3இல், தகவல் செல்லும் பாதைகளைத் தடித்த கோடுகளும், கட்டுப்பாட்டுச் சமிக்ஞைகள் செல்லும் பாதைகளை உடைந்த கோடுகளும் காட்டுகின்றன.



படம் 2 - 3 கணிப்பொறியின் அமைப்பு

நினைவில் நிற்க

1. ஒவ்வொரு கணிதச் செயல்பாட்டிற்கும், தர்க்கச் செயல்பாட்டிற்கும் தேவையான சுற்றுகளை உள்ளடக்கியது, கணித தர்க்கச் செயலகம்.

2. மற்ற எல்லாப் பகுதிகளையும் கட்டுப்படுத்தி, நாம் கொடுத்த கணக்கைச் செய்து முடிக்க உதவுவது கட்டுப்பாட்டகம்.

3. கட்டுப்பாட்டகம், கணித தர்க்கச் செயலகம் மற்றும் சில பதிவிடங்கள் சேர்ந்தது மையச் செயலகம்.

4. தகவல்கள், கணக்கிடும் முறை, இயக்க முறை இவற்றை வைக்கும் இடம் நினைவகம். நினைவகத்தில் உள்ள இடங்களுக்கு முகவரி எண்கள் உண்டு.

5. உள்ளீட்டகங்களும், வெளியீட்டகங்களும், கணிப்பொறியின் வெளி உலகத் தொடர்புச் சாதனங்கள்.

பயிற்சி வினாக்கள்

1. கணிப்பொறியின் அடிப்படைப் பாகங்களையும், அவற்றின் இணைப்பையும் ஒரு படம் வரைந்து விவரி.
2. நினைவகத்தில் ஒரு வார்த்தையை வைக்கும் முறையையும், எடுக்கும் முறையையும் விவரி.
3. பஸ் என்றால் என்ன?
4. சொல் நீளம் என்பதை விவரி.
5. K, M, B என்பவை எதைக் குறிக்கின்றன?
6. இரண்டு இருநிலை இலக்கங்களைக் கூட்டும்போது, வரும் கூட்டுத்தொகை மற்றும் அடுத்த இடத்திற்குச் செல்லும் தொகை இவற்றைக் கணக்கிட எப்படிச் சுற்றுகளை வடிவாக்கலாம் என்பதை விவரி.
7. கணித தர்க்கச் செயலகத்தின் செயல்பாட்டினை விளக்கு.
8. கட்டுப்பாட்டகத்தின் வேலை என்ன? அந்த வேலையை எப்படிச் செய்யும்?
9. மையச் செயலகத்தின் பாகங்களை விவரி.
10. பதிவிடம் என்றால் என்ன? ஒரு கணிப்பொறியில் சுமாராக எத்தனை பதிவிடங்கள் இருக்கும்?

2.2 உள் நினைவகங்கள்

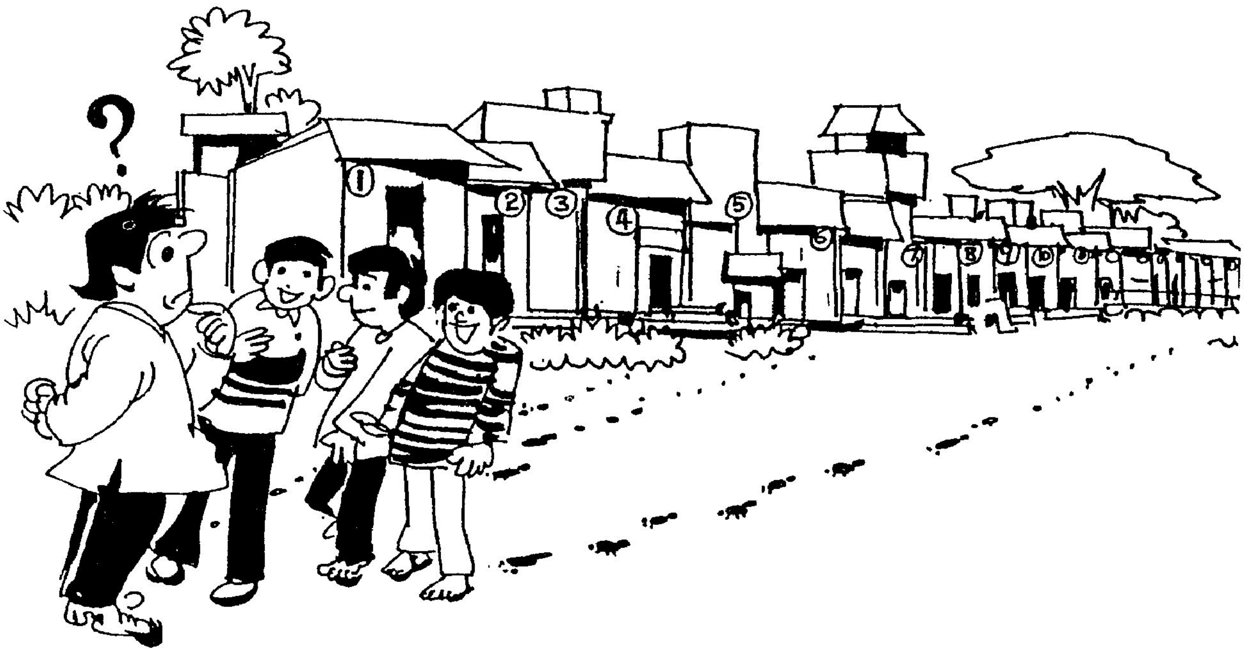
நம் நினைவகம் மூளை. இதில் ஓரளவுக்கு மேல் தகவல்களை வைக்க முடியாது. ஒருவர் இறக்கும்பேது அவர் நினைவில் இருந்தவையும் அழிந்துவிடும். ஒருவருக்குத் தெரிந்த தகவல் மற்றவர்களுக்கு எந்த நேரத்திலும் (இவர் உயிர் இழந்த பின்னரும்கூட) பயன்பட வேண்டுமானால் அதற்கு ஒரே ஒரு வழிதான் உண்டு. அந்தத் தகவலை ஏதாவது ஒரு இடத்தில் எழுதி வைப்பது தான் அது. கல், களிமண் பலகை, தாமிரப் பட்டயம், ஓலைச்சுவடி, காகிதம் போன்றவை மனிதன் எழுதிவைக்கப் பயன்படுத்திய சாதனங்கள்.

மனிதனைப் போலவேதான் இந்தக் கணிப்பொறியும். சென்ற பாடத்தில் நாம் பார்த்தது கணிப்பொறியின் **உள் நினைவகம்** (Internal Memory). இது குறிப்பிட்ட அளவே உள்ளது. கணிப்பொறியை அணைத்தவுடன், மின்சாரம் நின்றவுடன், இதில் உள்ள செய்திகள் அழிந்துவிடும். இவ்வகை நினைவகம் **அழியும் நினைவகம்** (Volatile Memory) எனப்படும். மற்றக் கணிப்பொறிகள் அல்லது அதே கணிப்பொறி எந்த நேரத்திலும் பயன்படுத்த வேண்டுமானால், தகவல்களைக் கணிப்பொறிக்கு வெளியே உள்ள **வெளி நினைவகத்தில்** (Enternal Memory) வைப்பது அவசியம். உள்நின்னைவகங்கள் பற்றி இந்தப் பாடத்திலும், வெளிநினை வகங்கள் பற்றி அடுத்த பாடங்களிலும் பார்ப்போம்.

IBM 1620 போன்ற பொறிகளில் உள்ள உள் நினைவகம் சிறு காந்த வளையங்களால் (Magnetic core) ஆனது. ஒவ்வொரு வளையமும் அதில் உள்ள காந்தத்தின் திசையைப் பொறுத்து 0 அல்லது 1 என்பதை நினைவில் வைக்கும். இவ்வகை நினைவகத்திற்கு இடம் அதிகம் தேவை. அதனால் இப்போது இவ்வகை பயன்படுவதில்லை.

இன்றைய கணிப்பொறிகளில் பெரிதும் பயன்படுவது, சில்லுகளில் வரும், மின்சுற்றுகளால் ஆன, மின்னணு நினைவகங்கள். ஒரு சில்லிலேயே 32KB, 64KB, 1MB போன்ற அளவுகளில் இப்போது கிடைக்கின்றன. இவற்றின் விலையும் குறைவு. இவற்றில் பலவற்றை வரிசையாக வைத்துத் தேவையான அளவு நினைவகத்தை ஏற்படுத்திக் கொள்ளலாம். **குறை கடத்திகளால்** (Semi Conductors) உருவாக்கப்படுவதால் இவை **குறைகடத்தி நினைவகம்** (Semi Conductor Memory) எனப்படும்.

நினைவகத்திலிருந்து ஒரு வார்த்தையைப் பெற, முகவரி எண்ணைக் கொடுத்துக் கேட்க வேண்டும் என்று பார்த்திருக்கிறோம். முகவரியைத் தேடி வார்த்தையைக் கொண்டுவர எவ்வளவு நேரமாகும்? முகவரி சிறிய எண்ணாக இருந்தால் விரைவாகவும், பெரிய எண்ணாக இருந்தால் அதிக நேரமெடுத்தும் தேட வேண்டுமா? இல்லை! எந்த முகவரியாக இருந்தாலும் தேடி எடுக்கும்



"அதெப்படி எந்த வீட்டுக்காரரைக் கூப்பிட்டாலும் ஒரே அளவு நேரத்திலே வர்ரீங்க?"

நேரம் ஒன்றுதான்! நினைவகத்தின் சுற்று அவ்வாறு வடிவமைக்கப்பட்டிருக்கிறது. அதனால் இது நேரடி அணுகுமுறை நினைவகம் (RAM - Random Access Memory) என்றும் பெயர்கொண்டது.

படிக்க மட்டும் நினைவகம் (ROM - Read Only Memory)

சில்லுகளாக வரும் குறைகடத்தி நினைவகங்களில், RAM தவிர ROM, PROM, EPROM, EAPROM என்று பல வகைகள் உண்டு. இவற்றில் எழுதி வைக்கப்படும் தகவல்கள் மின்சாரம் நிறுத்தப்பட்டாலும் அழியாது. அதனால் இவை அழியாவகை (Non volatile). சாதாரணமாக, கணிப்பொறிகளால் இவற்றில் எழுத முடியாது; படிக்க மட்டும்தான் முடியும். அதனால்தான் இவை எல்லாவற்றின் பெயரிலும் படிக்க மட்டும் (Read only) என்று வருகிறது.

சில்லை உருவாக்கும்போது மட்டும் எழுதிவைத்து, பிறகு எப்போதும் அழிக்கவும் திரும்ப எழுதவும் முடியாத வகை ROM (Read Only Memory) எனப்படுவது. கணிப்பொறியின் அடிப்படைச் செயல்கள் சிலவற்றைச் செய்யும் கட்களைத் தொடர்களை எழுதிவைக்க இவ்வகை நினைவகங்கள் பயன்படு கின்றன. இக்கட்டளைத் தொடர்களைப் படித்துச் செயலாற்ற வேண்டுமே தவிர அவற்றை மாற்றக்கூடாது. இம்மாதிரியான மென்பொருள்கள் **நிலைப்பொருள்** (Firmware) எனப்படும். இந்த வகை நினைவகங்கள் கணிப்பொறியின் உள்நினைவகமாகப் பயன்படும்.

நமக்குத் தேவையானவற்றை, ஒரு தனி சாதனத்தைப் பயன்படுத்தி ஒருமுறை எழுதலாம்; பிறகு படிக்கலாமே தவிர. அழித்து எழுத முடியாது. இந்த வகை நினைவகச் சில்லுகள் PROM - Programmable Read Only Memory *எனப்படும். அடுத்த வகைகளான* EPROM (Erasable and Programmable Read Only Memory), EAPROM (Electrically Alterable and Programmable Read Only Memory) *என்பவற்றில் அழித்து அழித்து திரும்ப எழுத முடியும். ஆனால் இதற்குத் தனிச் சாதனங்கள் தேவை. கணிப்பொறியால் இவற்றைப் படிக்க மட்டும் முடியும்.*

நினைவில் நிற்க

1. உள் நினைவகங்களில் RAM அழியும் வகையைச் சேர்ந்தது.

2. PROM, EPROM, EAPROM போன்றவை அழியாவகையைச் சேர்ந்த உள்நினைவகங்கள்.

3. நிறுவனங்கள் சில்லுகளில் பதித்துத் தரும் மென்பொருள்கள் நிலைப்பொருள்கள் எனப்படும்.

பயிற்சி வினாக்கள்

1. உள் நினைவகங்களின் வகைகள் எவை?

2. ROM, PROM, EAPROM என்பவற்றின் விரிவைக் கூறு.

3. RAMக்கும் ROMக்கும் உள்ள வித்தியாசங்கள் இரண்டைக் கூறு.

4. உள்நினைவகங்கள் எல்லாம் அழியும் வகையைச் சேர்ந்தது என்பது சரியா அல்லது தவறா என்று கூறுக.

2.3 வெளிநினைவகம் - தகடுகள்

வெளி நினைவகங்கள்

துளை அட்டைகள், துளைக் காகிதம், காந்த உருளை, காந்தப் பட்டை போன்றவை ஆரம்ப காலத்தில் தோன்றிய வெளி நினைவகங்கள். இன்று அதிகம் பயன்படுவது **நெகிழ் தகடு** (Floppy disc), **வன்தகடு** (Hard disc), **காந்த நாடா** (Magnetic tape), மற்றும் CDROM (Compact disc read only memory). இவற்றுள் CDROM என்பது லேசர் ஒளியால் இயக்கப்படும் தகடு. இதன் கொள்ளளவு பல நூறு மெகா பைட்டுகள். கலைக் களஞ்சியங்கள் போன்றவை இத்தகைய தகடுகளில் இப்போது கிடைக்கின்றன. இவற்றை அழித்து எழுத முடியாது. அழித்து எழுதும் வகைகள் சிலவும் வரத் தொடங்கியுள்ளன. நெகிழ்தகடு, வன்தகடுகளைப் பற்றி இந்தப் பாடத்திலும், காந்த நாடா பற்றி அடுத்த பாடத்திலும் பார்ப்போம்.

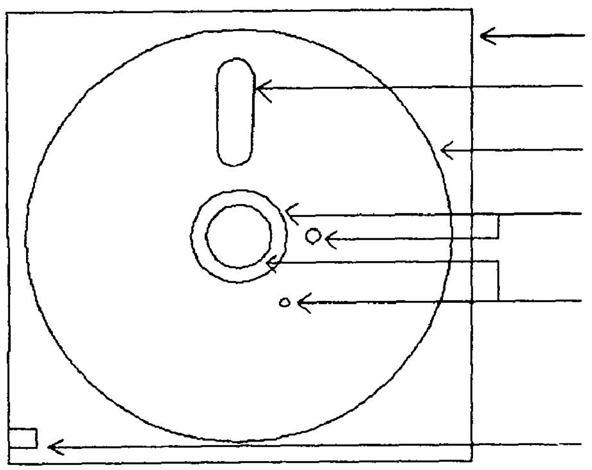
நெகிழ்தகடு

காந்தமேற்றக்கூடிய மேற்பரப்பைக் கொண்ட, வளையக்கூடிய வட்டத் தகடு இது. (ஆனால் வளைக்கக் கூடாது. வளைத்தால் கெட்டுப்போய்விடும்). இதில் ஒரே மையம் கொண்ட வட்டங்கள் பல இருக்கும். ஒவ்வொரு வட்டமும் பல **பகுதிகளாகப்** (Sector) பிரிக்கப்பட்டிருக்கும். ஒவ்வொரு பகுதியிலும் சாதாரணமாக 512 பைட்டுகளை எழுதலாம். ஒரு புள்ளியில் காந்தம் எந்தத் திசையை நோக்கி இருக்கிறது என்பது 0 அல்லது 1 என்பதைக் குறிக்கும். பெரும்பாலும் தகட்டின் இரண்டு பக்கங்களிலும் எழுதிப் படிக்கலாம்.

தகட்டின் விட்டம் 8", 5¼", 3½" என்று குறைந்துகொண்டே வருகிறது. ஆனால் அதன் கொள்ளளவு கூடிக் கொண்டே போகிறது. 5 ¼, 3 ½ தகடுகள் இன்று பெரிதும் பயன்படுகின்றன. இதிலும் பலவித அடர்த்திகள் உள்ளன. 360 KB, 720 KB, 1.2 MB, 1.44 MB என்று பல வகை. முன்பு வந்த நெகிழ்தகடு இயக்கிகள் (Floppy disc drives) ஒரு குறிப்பிட்ட அடர்த்தி உள்ள தகடுகளை மட்டுமே பயன்படுத்த முடியும். இப்போது வரும் இயக்கிகள் பல அடர்த்தித் தகடுகளையும் எழுதிப் படிக்கும் ஆற்றல் பெற்றவை.

ஒரு தகட்டில், அதன் கொள்ளளவைப் பொறுத்து, அதில் எத்தனை காந்த வட்டங்கள் உள்ளன, ஒரு வட்டத்தில் எத்தனை பகுதிகள் உள்ளன, ஒரு வட்டத்தின் பருமன் எவ்வளவு போன்றவை மாறுபடும். 360 KB தகட்டில் ஓர் அங்குலத்திற்கு 48 வட்டங்கள் இருக்கும். இது **இருமடங்கு அடர்த்தி** (Double density) எனப்படும். 1.2 MB தகட்டில் ஓர் அங்குலத்திற்கு 96 வட்டங்கள் இருக்கும். இது **உயர் அடர்த்தி** (High Density) எனப்படும்.

தகடு கெடாமலிருக்க, அது ஒரு பிளாஸ்டிக் உறைக்குள் இருக்கும். தகட்டை, உறையை விட்டு எடுக்காமலேயே எழுதிப் படிக்க வசதியாக உரையில் ஆரம்போல் ஒரு நீளமான துளை இருக்கும். ஒவ்வொரு வட்டத்திலும் முதல் பகுதி எங்கு ஆரம் பிக்கிறது என்பதைக் குறிக்க, 5¼" தகட்டில் ஒரு சிறு துளை இருக்கும். இத்துளை வெளியில் தெரிவதற்காக உறையிலும் ஒரு சிறு துளையிருக்கும். இயக்கியில் தகடு சரியாகப் பொருத்துவதற்காக, தகட்டின் நடுவில் ஒரு பெரிய துளை இருக்கும். ஆக மொத்தம், தகட்டில் இரண்டு துளைகளும், உறையின் ஒரு பக்கத்தில் மூன்று துளைகளும் இருக்கும். படம் 2-4 இவ்வமைப்பைக் காட்டுகிறது. உறையின் ஓர் ஓரத்தில் சிறு செவ்வக வடிவ வெட்டு ஒன்று இருக்கும். இவ்வெட்டு திறந்திருந் தால்தான் இத்தகட்டில் எழுத முடியும். எழுதுவதைத் தவிர்க்க, இவ் வெட்டினை, கனமான சிறு காகித அல்லது பிளாஸ்டிக் துண்டினால் மூடிவிட வேண்டும்.

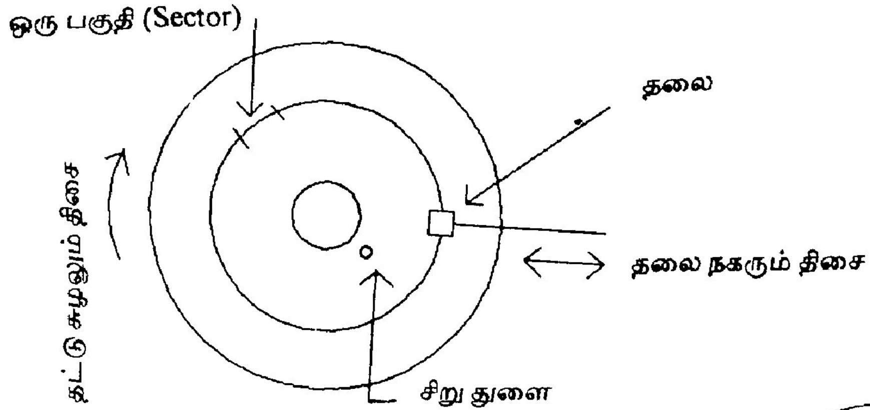


படம் 2 - 4 நெகழ் தகடு

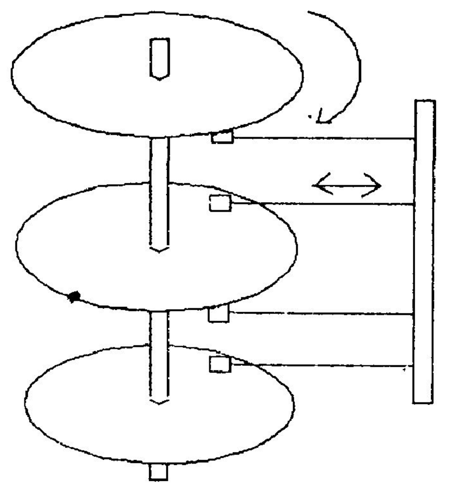
செயல்முறை

எழுத அல்லது படிக்க வேண்டியபோது, தகடானது ஒரே வேகத்தில், நிமிடத்திற்கு 300 அல்லது 360 முறைகள் (300 or 360 rpm) சுற்றும். எழுதிப் படிக்கும் தலைகள் பக்கத்திற்கு ஒன்றாக, இரண்டு உண்டு. தலைகள் ஓர் ஆரத்தின் திசையில் உள்ளேயும் வெளியேயும் நகரும் (படம் 2-5). முதலில் தலையானது தேவையான வட்டத்திற்கு மேல் வந்து நிற்கும். பிறகு, தட்டு சுழலும்போது தேவையான பகுதி (Sector) தலைக்குக் கீழ் வரும்போது அந்தப் பகுதியை முழுவதுமாகப் படித்து அல்லது எழுதி விடும்.

இதில் நகரும் பாகங்கள் உள்ளதால், CPU போன்ற மின்னணுச் சாதனங் களுடன் ஒப்பிடும்போது, இது மிகவும் மெதுவாகச் செயல்படும். நுண்கணிப் பொறிகளில் மிகவும் பயன்படும், விலை குறைந்த வெளிநினைவகம் இந்த நெகிழ்தகடுகள்தான்.



படம் 2 - 5 நெகிழ் தகடு இயங்கும் முறை



படம் 2 - 6 வன்தகடு

வன்தகடு

வன்தகடு என்று இதை ஒருமையில் குறிப்பிட்டாலும், இதற்குள் பல தகடுகள் உள்ளன. இட்டிலித் தட்டுகள் போல் பல தகடுகள் ஒன்றின் மீது ஒன்றாக அடுக்கி இணைக்கப்பட்ட தொகுதி இது. இந்தத் தகடுகள் வளையாது. இதில் உள்ள தகடுகளில், மேல் தகட்டின் மேற்புறத்தையும், கீழ்த் தகட்டின் கீழ்ப்புறத்தையும் விட்டுவிட்டு மீதிப் பக்கங்கள் மட்டும் பயன்படுத்தப்படும். அதனால் 6 தகடுகள் உள்ள தொகுதியில், (6 x 2) - 2 = 10 பக்கங்கள் பயன்படும். பத்து எழுதிப் படிக்கும் தலைகள் இணைந்து செயல்படும். படம் 2-6 இந்த அமைப்பை விளக்கும். இந்தத் தகடுகளின் அமைப்பும், செயல்பாடும் நெகிழ் தகட்டைப் போலவே இருக்கும். சில வித்தியாசங்கள் மட்டும் உண்டு.

1. இத்தகடுகள் பெட்டி போன்ற உறையினுள் வைக்கப்பட்டிருக்கும்.
2. இது நிமிடத்திற்கு 3600 முறைகள் சுற்றும். அதனால் வேகம் சற்று அதிகம்.
3. தகடுகள் வளையாது.
4. தகடுகளின் கொள்ளளவு 10MB, 20, 40, 80MB என்று ஆரம்பித்து 270MB, 600 MB, 1GB, 2GB என்று மேலே செல்லும்.

இந்த வன்தகடுகளில் இரண்டு வகைகள் உண்டு.

1. பெட்டிக்குள்ளேயே தகடுகளும், அவற்றைப் படிக்கும் தலைகளும், இயக்கியும் வைக்கப்பட்டு, காற்றுப் புகாமல் அடைக்கப்பட்டிருக்கும். இது நுண்கணிப் பொறிகளில், கணிப்பொறியின் பெட்டிக்குள்ளேயே நிலையாகப் பொருத்தப்பட்டிருக்கும். கணிப்பொறிப் பெட்டிக்கு உள்ளே இருந்தாலும் இது ஒரு வெளி நினைவகம்தான். நுண்கணிப் பொறிகள் அலசும் தகவல்களின் அளவு குறைவாதலால், இந்தத் தகட்டின் தொகுதியை மாற்ற வேண்டிய அவசியம் ஏற்படாது.

2. பெரிய கணிப்பொறிகளில் பல்லாயிரக்கணக்கான மெகாபைட் அளவில் நினைவிடங்கள் தேவையிருக்கும். அப்போது ஒரு வன்தகடு போதாது. பல தகடுகளைப் பயன்படுத்த வேண்டியிருக்கும். அதனால், அதன் இயக்கி, ஒரு குளிர்சாதனப் பெட்டி அளவில் தனியாக இருக்கும். இதில் தேவையான தகட்டுப் தொகுதியைப் போட்டு இயக்கலாம். தகட்டுத் தொகுதிகளைத் தனியாக எடுத்து வைக்கவும் முடியும்.

நன்மை

நெகழ் மற்றும் வன்தகடுகளைப் பயன்படுத்துவதில் ஒரு பெரும் நன்மை உண்டு. நமக்குத் தேவையான தகவல், தகட்டில் எங்கே இருந்தாலும், குறைந்த நேரத்தில் அதைப் பெற முடியும்.

தலை உள் வட்டத்திலிருந்து வெளி வட்டத்திற்கு வரும் நேரம் + தட்டு ஒரு முறை சுற்ற ஆகும் நேரம் - இதுதான் எந்தத் தகவலையும் படிக்க ஆகும் அதிகபட்ச நேரம். அதனால், இவை **நேரடி அணுகுமுறைச் சாதனங்கள்** (Random Access Devices) எனப்படும்.

நினைவில் நிற்க

1. தகடுகள் நேரடி அணுகு முறையைச் சேர்ந்தவை; அவை வெளி நினைவகங்கள்.

2. வன்தகட்டில், பல தகடுகள் இருக்கும். அதிக வேகத்தில் சுற்றும். கொள்ளளவு பல மெகா பைட்டுகள்.

3. ஒவ்வொரு பக்கத்திற்கும் ஒரு எழுதும்/படிக்கும் தலை உண்டு.

பயிற்சி வினாக்கள்

1. நெகிழ்தகட்டின் அமைப்பையும், செயல்பாட்டையும் விவரி.

2. வன்தகட்டின் அமைப்பையும், செயல்பாட்டையும் விவரி.

3. வன்தகடுகளில் உள்ள இரு வகைகள் யாவை?

4. நெகிழ்தகட்டிற்கும், வன்தகட்டிற்கும் உள்ள வித்தயாசங்களைப் பட்டியலிடு. அவற்றின் ஒற்றுமைகளையும் கூறு.

5. நெகிழ்தகட்டின் சாதக பாதகங்களைக் கூறு.

6. வன்தகடு ஒரு

அ. உள் நினைவகம் ஆ. வெளி நினைவகம்

இ. இரண்டுமே ஈ. இரண்டும் இல்லை.

7. வன்தகட்டில்

அ. ஒரு தகடு மட்டும் இருக்கும்

ஆ. 4 தகடுகள் இருக்கும்.

இ. பல தக்டுகள் இருக்கும்.

8. 5 ¼ " நெகிழ்தகட்டில் இருக்கும் துளைகளின் எண்ணிக்கை

அ. 1 ஆ. 2 இ. 3 ஈ. 4

9. வன்தகட்டினை இயக்கியை விட்டுத் தனியாக எடுத்து வைக்க முடியும்.

அ. தவறு ஆ. சரி இ. சில வகைகளில் மட்டும் முடியும்.

10. நெகிழ்தகடு சாதாரணமாகச் சுற்றும் வேகம்

அ. 3000 அல்லது 3600 rpm ஆ. 500 அல்லது 640 rpm

இ. 1000 அல்லது 1024 rpm ஈ. 300 அல்லது 360 rpm

11. வன்தகடுகளின் வேகம், நெகிழ்தகடுகளின் வேகத்தைப் போல், சுமார்.

அ. 2 பங்கு ஆ. 4 பங்கு

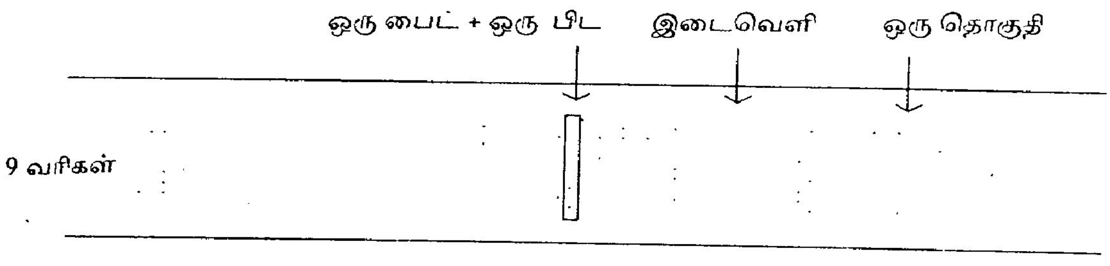
இ. 10 பங்கு ஈ. 20 பங்கு

2.4 வெளி நினைவகம் - காந்த நாடா

தகவல்களை **வரிசையாக அணுகும் முறையில்** (Sequential access) அமைந்த வெளி நினைவகம் காந்த நாடா. நாம் பாட்டுகள் கேட்கும் ஒலி நாடாக்களும், திரைப்படம் பார்க்கும் ஒளி நாடாக்களும், தகவல்களை வைக்கும் இந்தக் காந்த நாடாக்களும் ஒரே வகைதான். ஒலி நாடாவில், பாட்டுகளை ஒவ்வொன்றாக அடுத்தடுத்து பதிவு செய்கிறோம். ஒரு பாட்டுக்கும் அடுத்த பாட்டுக்கும் இடையில் சிறிது இடைவெளி விட்டு பதிவு செய்வது வழக்கம். இதில் ஒரு நன்மை உண்டு. சில பாடல்களை வேகமாகக் கடந்து சென்று, நமக்குப் பிடித்த பாடலை மட்டும், வேறெந்தப் பாடலுடனும் ஒட்டாமல் கேட்டு ரசிக்க முடியும். **ஒலிப் பேழைகள்** (Audio cassettees) வருவதற்கு முன்னால், திரைப்படப் ஃபிலிம் சுருள்போல் பெரிய உருளைகளில் இந்த ஒலி நாடாக்கள் வந்தன.

தகவலை வைக்கும் காந்த நாடாக்கள் இதேபோல் உருளைகளில் இருக்கும். இதை இயக்கும் சாதனங்கள் அலமாரி அளவில் இருக்கும். ஏற்கனவே பார்த்தது போல் ஒரு காந்தப் புள்ளியின் திசையானது 0 அல்லது 1 என்பதைக் குறிக்கும். எளிதில் படிப்பதற்காக, தவறுகள் நேராதிருப்பதற்காக, தகவல்கள் குறிப்பிட்ட அளவுள்ள தொகுதிகளாக மாற்றப்படும்.

ஒவ்வொரு தொகுதிக்கு இடையேயும் குறிப்பிட்ட அளவு இடைவெளி விட்டு பதிவு செய்யப்படும். படம் 2.7 இதை விளக்கும்.



படம் 2 - 7 காந்த நாடா

இதன் சாதக அம்சம் - குறைந்த விலையில் நிறைய இடம். இதன் பாதக அம்சம் - வரிசை அணுகுமுறை. நாடாவின் மறு கோடியில் உள்ள தகவலைப் படிக்க நாடா முழுவதையும் கடந்தாக வேண்டும். இதற்கு நிறைய நேரம் (பல வினாடிகள்) பிடிக்கும்.

இந்த காந்த நாடாக்கள் பெரிய அளவு கணிப்பொறிகளில்தான் பெரிதும் பயன்படும். ஒலிப்பேழையில், ஒலிப்பதிவுக் கருவியைக் கொண்டே தகவல்களையும் பதிவு செய்ய முடியும்! குறைந்த விலை காரணமாக, ஒலிப்பேழைகளை

ஆரம்பகாலத்தில் நுண்கணிப்பொறிகளுடன் பயன்படுத்தினார்கள். ஆனால் இந்த ஒலிப்பதிவுக் கருவிகள் நம்பகமாகச் செயல்படாததால் இப்போது எவரும் அதை விரும்புவதில்லை. ஒலிப்பேழைகளைப் போல் தகவல் நாடாப் பேழைகளும் இப்போது வருகின்றன. விரைவாகச் செயல்படுவதாலும், அதிகக் கொள்ளளவு உள்ளதாலும், தகவல்களைப் பத்திரப்படுத்தி வைக்க இவை பெரிதும் பயன் படுகின்றன.

ஒன்பதாம் பிட்

காந்த நாடாவில் சாதாரணமாக 9 காந்தப் புள்ளி வரிகள் இருக்கும். இந்த வரிகள் நாடாவுக்கு இணையாக இருக்கும். நாடாவுக்குக் குறுக்கு வெட்டில் இருக்கும் ஒவ்வொரு வரிசையிலும் 9 பிட்டுகள் இருக்கும். 8 பிட்டுகள் சேர்ந்தது ஒரு பைட். இந்த ஒன்பதாம் பிட் எதற்கு? ஒரு பைட்டைப் பதிவு செய்து வைத்து, பிறகு அதைப் படிக்கும்போது ஏதகவது தவறு நேர்ந்திருக்கிறதா என்பதை ஓரளவு கண்டுபிடிக்க உதவும் இந்த பிட். ஒரு பைட்டைப் பதிவு செய்யும்போது, அதில் உள்ள ஒன்றுகள் ஒற்றைப் படையில் இருந்தால் இந்த பிட் 1 எனவும், இரட்டைப் படையில் இருந்தால் இந்த பிட் 0 எனவும் எழுதிவைக்கப்படும். இதனால், மொத்தம் ஒன்பது பிட்டுகளிலும் உள்ள ஒன்றுகளின் எண்ணிக்கை இரட்டைப் படையில் வர வேண்டும். ஒவ்வொரு முறை படிக்கப்படும்போதும் இது சரிபார்க்கப்படும். ஏதாவது காரணங்களினால், ஒரு பிட்டில் தவறு ஏற்பட்டிருந் தால், அதாவது ஒரு சுழியானது ஒன்றாகவோ, அல்லது ஒரு ஒன்று சுழியாகவோ மாறியிருந்தால், மொத்தத்தில் உள்ள ஒன்றுகளின் எண்ணிக்கை ஒற்றைப் படையாக ஆகிவிடும். இதனால் படிப்பதில் ஏதோ தவறு நேர்ந்திருக்கிறது என்பது தெரிந்து விடும். ஆனால் தவறு எங்கு நடந்திருக்கிறது என்பது தெரியாது. இரண்டு இடங் களில் தவறு நடந்திருந்தால் இம்முறையில் கண்டுபிடிக்க முடியாது. உதாரணமாக 1010 1110 என்னும் பைட் 1010 1110 1 என்று எழுதிவைக்கப்படும். இதைப் படிக்கும் போது 1011 1110 1 என்று வந்தால், இதில் ஒற்றைப்படை ஒன்றுகள் இருப்பதால் எங்கோ தவறு நடந்திருக்கிறது என்று தேரியும். இதுவே படிக்கும்போது 1001 1110 1 என்று வந்தால், தவறு நடந்திருப்பது தெரியாது. ஆனால் இரண்டு தவறுகள் நேர்வதற்கு வாய்ப்பு மிகவும் குறைவு. கணிப்பொறி தவறு செய்யலாம். ஆனால் கண்டுபிடித்து திருத்திக்கொள்ளாமல் போவதற்கு வாய்ப்பு மிகவும் குறைவு.

மொத்தத்தில் இரட்டைப்படை ஒன்றுகள் இருக்க வேண்டும் என்பதால் இம்முறை **இரட்டைப்படை சரிபார்க்கும் முறை** (Even parity check) எனப்படும். இதே போல் **ஒற்றைப் படை சரிபார்க்கும் முறையையும்** பயன்படுத்தலாம்.

நினைவில் நிற்க

1. நாடாக்களில் தகவல்களை வரிசையாக மட்டுமே அணுகமுடியும். அதனால், சில சமயங்களில் தகவல் பெற அதிக நேரமாகும்.

2. தகவல்கள் தவறில்லாமல் படிக்கப்பட்டதா என்பதைச் சரிபார்க்க இரட்டைப்படை சரிபார்க்கும் முறை பயன்படுகிறது.

3. கணிப்பொறி தவறு செய்யக்கூடும். ஆனால் இது மிக மிக அரிது.

பயிற்சி வினாக்கள்

1. காந்த நாடாவில் தகவல் பதிவு செய்யும் விதத்தை விவரி.
2. காந்த நாடாவின் சாதக, பாதக அம்சங்களைக் கூறு
3. இரட்டைப்படை சரிபார்க்கும் முறையை ஓர் உதாரணத்துடன் விளக்கு.
4. அடிக்கடி பயன்படும் தகவல்களை சேர்த்து வைக்க சிறந்த சாதனம்

   அ. காந்த நாடா ஆ. துளை அட்டைகள்

   இ. வுன்தகடு ஈ. ஒளிநாடா

5. மிக அதிக அளவிலான தகவல்களை, மிகக்குறைந்த செலவில் சேர்த்து வைக்க சிறந்த வழி

   அ. காந்த நாடா ஆ. நெகிழ்தகடு

   இ. வன்தகடு ஈ துளை அட்டைகள்.

6. காந்த நாடாவில் ஒரு தகவலை அணுக எடுத்துக்கொள்ளும் நேரம் அந்தத் தகவல் பதிவாகியிருக்கும் இடத்தைப் பொறுத்தது.

   அ. ஆம் ஆ. இல்லை.

2.5 உள்ளீட்டகங்கள்

தகடுகளும், காந்த நாடா இயக்கிகளும், உள்ளீட்டகம் மற்றும் வெளியீட்டகம் என்று இரண்டு விதங்களிலும் பயன்படும். இவற்றைப் பற்றி ஏற்கனவே பார்த்து விட்டோம். **சாவிப்பலகை** (Key Board), **ஒளியியல் படிப்பான்** (OCR - Optical character reader), **காந்த எழுத்துப் படிப்பான்** (MICR - Magnetic ink character reader), **சுண்டெலி** (Mouse), **மின்னணுப் பலகை** (Electronic tablet) என்ற உள்ளீட்டகங்களைப் பற்றி இந்தப் பாடத்தில் அறிந்துகொள்வோம். **ஒளிப்பேனா** (Light pen), **ஜாய் ஸ்டிக்** (Joy stick), **டிராக்கர் பந்து** (Tracker ball), **விரல் சக்கரங்கள்** (Thumb wheels) போன்றவை மற்றும் சில உள்ளீட்டகங்கள். இப்போது புதிதாக வரும் ஓர் உள்ளீட்டகம் **தொடுதிரை** (Touch screen) என்பது. படத்திரையில் ஓர் இடத்தைத் தேர்ந்தெடுக்க, அந்த இடத்தை விரலால் சிறிது அழுத்த வேண்டும். அவ்வளவுதான். ஒலியலைகளை உள்ளீடாகப் பெறும் **மைக்குகளும்** (Mike) இன்று பிரபலமாகி வருகின்றன. அது மட்டுமா? நம் கண்கள், திரையில் எந்த இடத்தைப் பார்க்கின்றன என்பதைக் கண்டுபிடித்து, அதையே உள்ளீடாகக் கொள்ளும் சாதனங்களும் வரப்போகின்றன.

சாவிப்பலகை

101 சாவிகள் கொண்ட பலகை இன்று பிரபலமாக உள்ளது. சாதாரண ஆங்கில தட்டச்சுப் பொறியில் உள்ள சாவிகள் எல்லாம் இதிலும் உண்டு. மேலும், **சார்புசாவிகள்** (Function keys), **திசைச்சாவிகள்** (Cursor keys/arrow keys), மற்றும் சில **சிறப்புச் சாவிகள்** (Special Keys) என்பவையும் உண்டு. சார்பு சாவிகள் மொத்தம் 12 இருக்கும். இந்தச் சாவி ஒவ்வொன்றும், ஒரு குறிப்பிட்ட செயலைச் செய்யப் பயனாகும். உதாரணமாக, பேசிக் மொழிமாற்றியைப் பயன்படுத்தும் போது F1 என்ற சாவி ஒரு கோப்பினைக் கொண்டு வரவும், F2 என்பது கட்டளை நிரலைச் செயல்படுத்தவும் உதவும். கணிப்பொறியானது இயக்க அமைப்பின் (Operating system) ஆதிக்கத்தில் இருக்கும் போது, இதே சார்பு சாவிகள் வேறு செயல்களைச் செய்யும். இப்படிச் சூழ்நிலைக்கு ஏற்றாற்போல் இந்தச் சார்பு சாவிகளின் செயல்பாடும் மாறுபடும்.

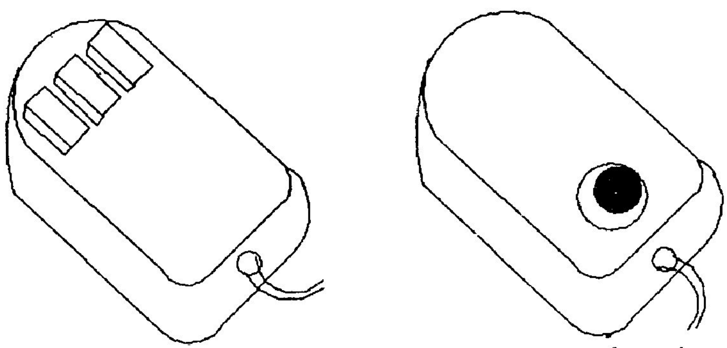
திரையில் தோன்றும் **இடம்சுட்டியை** (Cursor) நான்கு திசைகளிலும் நகர்த்தப் பயன்படும் சாவிகள் **திசைச்சாவிகள்** எனப்படும். இவை நான்கு அம்புக்குறி களால் (← → ↑ ↓) குறிக்கப்பட்டிருக்கும்.

தட்டச்சுப்பொறியில், **மாற்று** (Shift) சாவியுடன் இணைந்து இன்னொரு சாவியை அழுத்தும்போது, எழுத்து மாறி வரும். அதே போல் கன்ட்ரோல்

(CTRL - Control Key) அல்லது ஆல்ட்டர் (ALT - Alter Key) சாவியுடன் இணைந்து வேறொரு சாவியை அழுத்தும் போதும் புதிய குறியீடுகள் தோன்றும். இந்தக் கன்ட்ரோல், ஆல்டர், டிலீட் (Delete) போன்ற சாவிகள் சிறப்புச் சாவிகள் எனப்படும்.

சுண்டெலி (Mouse)

திரையில் ஒரு குறிப்பிட்ட புள்ளியை அல்லது இடத்தைத் தேர்ந்தெடுக்கப் பயன்படும் ஒரு சிறு சாதனம் இந்த சுண்டெலி என்னும் சிறுபொறி (படம் 2.8). இது கைக்கு அடக்கமாக இருக்கும். இதன்மேல் கையை வைத்துக்கொண்டு, இதை ஒரு சமதளத்தின் மேல் நகர்த்த வேண்டும். அப்போது, இதன் அடிப்புறத்தின் இருக்கும் ஒரு சிறு கோளம் நகரும். இதைப் பொறுத்து, திரையில் தோன்றும் இடம்சுட்டியும் நகரும். தேவையான இடத்திற்கு இடம்சுட்டி வந்தவுடன், சுண்டெலியின் மேற்பக்கம் இருக்கும் மூன்று சாவிகளில் ஒன்றை அழுத்தி, குறிப்பிட்ட இடத்தைத் தேர்ந்தெடுக்கலாம்.

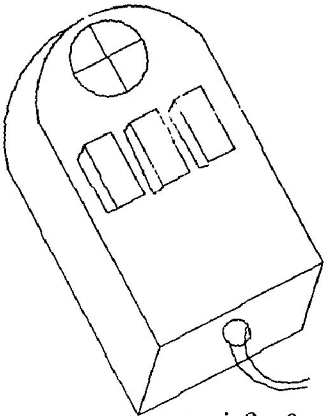


படம் 2 - 8 சுண்டெலி

மின்னணுப் பலகை (Electronic Tablet)

திரையில் படங்களை உருவாக்க இந்தப் பலகை உதவுகிறது. பலகையும் திரையும் சமமாகப் பாவிக்கப்படும். பலகையில் உள்ள ஒரு புள்ளியைத் தேர்ந்தெடுத்தால், அதற்குச் சமமான புள்ளி திரையில் தேர்ந்தெடுக்கப்படும். புள்ளியை தெரிவு செய்ய, சுண்டெலி போன்ற, கைக்கு அடக்கமான **பக்** (Puck) என்ற வில்லை (படம் 2.9) பயன்படும். இதில் உள்ள ஒளிபுகும் பிளாஸ்டிக் வில்லையின் நடுவில் உள்ள + குறியின் மையம், தேர்ந்தெடுக்கும் புள்ளியைக் குறிக்கும். சுண்டெலியில் இருப்பது போல் இதிலும் சாவிகள் உண்டு.

காகிதத்தில் ஏற்கனவே வரையப்பட்ட ப்டங்களை, அளவு மாறாமல், அப்படியே எளிதாக உள்ளீடு செய்ய இப்பலகை பெரிதும் பயன்படுகிறது. இதற்கு, வரைந்த காகிதத்தை பலகையின் மேல் நகராமல் பொருத்திவிட்டு, பிறகு



படம் 2 - 9 பக்

தேவையான புள்ளிகளை பக் மூலம் தெரிவு செய்ய வேண்டும். இந்தப் புள்ளிகளை இணைப்பதன் மூலம் தேவையான படத்தை உருவாக்கலாம். CAD (Computer Aided Design) எனப்படும் **கணிப்பொறி உதவிய வடிவமைப்பில்** மின்னணுப் பலகை அதிகம் பயன்படுகிறது. பக் என்பது பனித்தரை ஹாக்கி விளையாட்டில் பயன்படும் தட்டையான பந்து.

கணிப்பொறியின் பார்வை

தகவல்களை நாம் ஒவ்வொன்றாக உள்ளீடு செய்வதற்குப் பதிலாக, கணிப்பொறியானது தன் ''கண்கள்'' மூலம் ''பார்த்துத்'' தெரிந்து கொண்டால் எவ்வளவு சுலபமாக இருக்கும்! கட்டளை நிரல் எழுதிய காகிதத்தைப் பார்த்து, நிரலை நினைவகத்தில் வைத்துக்கொண்டால் நமக்கு நிறைய நேரம் மிச்சமாகும். கணிப்பொறிக்குக் கண்களாகச் செயல்பட சில உள்ளீட்டுச் சாதனங்கள் ஏற்கனவே உள்ளன. **வீடியோ கேமரா, ஸ்கேனர்** (Scanner), **பட்டைவரிப் படிப்பான்** (Bar Code Reader), **காந்த எழுத்துப் படிப்பான்** என்பன இவை.

வீடியோ கேமரா

சாதாரணமாக இவை, திரையில் தோன்றும் காட்சிகளை, புள்ளிகளாக மாற்றி, அவற்றைக் காந்தநாடா போன்ற நினைவகங்களில் பதிவு செய்து வைக்கும்.

ஸ்கேனர்

ஒரு படத்தை ஒளிக் கற்றையால் வருடி, அந்தப் படத்தைக் கறுப்பு/வெள்ளை அல்லது வண்ணப் புள்ளிகளாக மாற்றி, நினைவகத்தில் வைக்கும் அல்லது திரையில் காண்பிக்கும் சாதனம் இது. கையில் எடுத்து இயக்கும் அளவு, முழுப் பக்கத்தையும் படிக்கும் அளவு என்று பல அளவுகளில் இது கிடைக்கிறது. படங்களைக் கணிப்பொறிக்குள் செலுத்த உதவும் சிறந்த சாதனம் இது.

வீடியோ கேமரா அல்லது ஸ்கேனர் மூலம் ஒரு படத்தை உள்ளீடு செய்யும்போது, அந்தப் படம் முழுவதும் புள்ளிகளாகவே உள்ளே செல்கிறது. அந்தப் படம் ஒரு செய்தித்தாளாக இருந்தால் அதில் என்ன எழுத்துக்கள்

இருக்கின்றன என்ற விபரம் உடனே தெரியாது. வெறும் புள்ளிகள்தான் நினைவில் இருக்கும். இந்தப் புள்ளிகளை ஆராய்ந்து,. அதிலிருந்து எழுத்துகளை அடையாளம் காண்பதற்குத் தனியாக மென்பொருள்கள் (கட்டளை நிரல்கள் - software) வேண்டும். படிக்கப்படும் எழுத்துகள் அச்சிடப்படாமல் கையினால் எழுதப்பட்டிருந்தால், அந்த எழுத்துகளை இனம் காண்பது மிகவும் கடினமான செயல் (கணிப்பொறிக்கு). (சில மாணவர்கள் எழுதும் விடைத்தாள்களைப் படிப்பது எவ்வளவு கடினம் என்பது ஆசிரியர்களுக்குத்தான் தெரியும்!) இதற்கான மென்பொருள்கள் தயாரிப்பது பற்றிய ஆராய்ச்சி இன்றும் தொடர்கிறது. கணிப்பொறி விரைவாக எழுத்துகளை இனம் காண்பதற்காக, சற்று வித்தியாசமான முறையில் எழுத்துகளை அச்சிடுவதும் உண்டு.

காந்த எழுத்துப் படிப்பான்

இப்போது வரும் வங்கிக் காசோலைகளில் பெரும்பாலும் வரிசை எண், வங்கி எண் போன்றவை காந்த மையினால் எழுதப்பட்டிருக்கும். எண்களாகத் தெரிவதால் நாமும் அதைப் படிக்க முடியும். காந்தமையை உணர்வதன் மூலம், விலைகுறைந்த படிப்பான்கள் மூலம், கணிப்பொறியும் அதை விரைவில் படிக்க முடியும்.

படம் **2 - 10** பட்டை வரிகள்

பட்டை வரிப்படிப்பான் (Bar Code Reader)

கடைகளில் விற்கப்படும் பொருள்களின் மேல் படம் 2.10 இல் இருப்பது போல், பலவித பருமன்களில் கோடுகளைப் பார்த்திருப்பீர்கள். இக்குறியீடு பட்டைக் குறியீடு (Bar Code) எனப்படும். பொருளின் பெயர், அளவு, தயாரிப்பாளர் பெயர் முதலியவை இந்தக் கோடுகளில் அடங்கும். இதைப் படிக்கும் பொறி, கண்ணிமைக்கும் நேரத்தில் இந்தச் செய்திகளைக் கணிப்பொறிக்கு அனுப்பும். இதைப் பார்த்ததும் கணிப்பொறியானது பொருளின் விலை முதலியவற்றைத் தன் நினைவகத்திலிருந்து எடுத்து, தேவையான விற்பனைச் சீட்டைத் தயாரித்துக் கொடுக்கும். விரைவாகவும் கொடுக்கும்.

நினைவில் நிற்க

1. சாவிப் பலகையில் சாதாரண சாவிகளுடன், சார்பு சாவிகள், திசைச்

சாவிகள் மற்றும் சிறப்புச் சாவிகளும் உண்டு. சார்பு சாவிகள் குறிப்பிட்ட செயல்களைச் செய்யப் பயனாகும்.

2. இடம்சுட்டியை நகர்த்த, திசைச் சாவிகள் பயன்படும்.

3. படங்கள் வரைய மின்னணுப் பலகையும், பக்கும் உதவும். கணிப்பொறி உதவிய வடிவமைப்பில் இவை பெரிதும் பயன்படும்.

4. வீடியோ கேமரா, ஸ்கேனர், பட்டைவரிப் படிப்பான், காந்த எழுத்துப் படிப்பான் முதலியன கணிப்பொறியின் கண்களாக விளங்குகின்றன.

பயிற்சி வினாக்கள்

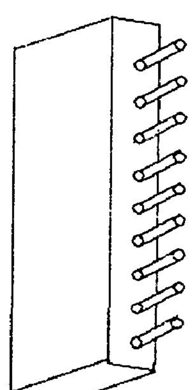
1. சாவிப் பலகையில் உள்ள சாவிகளை வகைப்படுத்து. அவற்றின் பயன்களைக் கூறு.
2. மின்னணுப் பலகை என்றால் என்ன? அதனுடன் இணைந்து பக் எப்படிச் செயல்படும்?
3. சுண்டெலியின் பயன் என்ன? அது எவ்வாறு செயல்படுகிறது?
4. கணிப்பொறிக்குப் பார்வை தரும் கண்களாகப் பயன்படும் சாதனங்கள் மூன்றினை விவரி.
5. MICR, OCR என்பவற்றை விவரி.
6. புதிய உள்ளீட்டுச் சாதனங்கள் இரண்டினைக் கூறு.
7. தொடுதிரை என்பதை விளக்கு.

2.6 வெளியீட்டகங்கள்

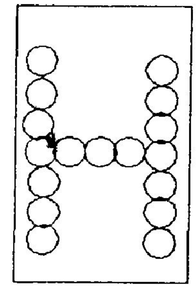
கணிப்பொறி கணித்துக் கொடுக்கும் விடைகளை வெளியிடுவதில் மூன்று முக்கிய வகைகள் உண்டு. ஒன்று - திரையில் காண்பித்தல். இதை நாம் பார்க்க முடியும். ஆனால் பிற்காலப் பயன்பாட்டிற்கு இது உதவாது. அடுத்தது, தகடுகள், நாடாக்கள் போன்றவற்றில் பதிவு செய்தல். இவ்வாறு பதிவு செய்தவற்றைக் கணிப்பொறியால் விரைந்து படிக்க முடியும். ஆனால், நம்மால் நேரடியாகக் கண்களால் படிக்க முடியாது. நாம் படிப்பதற்கு ஏற்ற வகையில் அச்சிட்டுத் தருவது மூன்றாம் வகை. இப்படி அச்சிட்ட **படி** Hard Copy எனப்படும். அச்சிடும் பொறிகளில் பல வகைகள் உண்டு. **புள்ளிவரிசை அச்சுப்பொறி, வரி அச்சுப் பொறி, லேசர் அச்சுப்பொறி**, மற்றும் **வரைவான்கள்** பற்றி இங்கு பார்ப்போம்.

புள்ளிவரிசை அச்சுப்பொறி (Dot Matrix Printer)

சிறிய அலுவலகங்களில் இன்று அதிகம் பயன்படுவது, விலை குறைந்த இந்தப் புள்ளிமுறை அச்சுப்பொறிகள்தான். நிமிடத்திற்குச் சுமார் 400 எழுத்துகள் வரை எழுதும் ஆற்றல் பெற்றவை. இதில் 9 புள்ளி வகை. 24 புள்ளி வகை என இரு வகைகள் உண்டு. இதில் முதல் வகையில், அச்சடிக்கும் தலையில் 9 மழுங்கிய ஊசிகள் இருக்கும். (படம் 2.11) ஓர் ஊசி முன்னால் வந்து நாடாவை அழுத்தும்போது காகிதத்தில் ஒரு புள்ளி பதிவாகிறது. ஒரே சமயத்தில் 8 ஊசிகள் வரை (மேல் 8 அல்லது கீழ் 8) தேவையான ஊசிகளை மட்டும் இயக்கலாம். ஒவ்வொரு முறையும் எட்டு புள்ளிகளால் ஆன ஒரு செங்குத்துக் கோட்டை அச்சிட்டதும், தலையானது சிறிது வலது பக்கம் நகர்ந்து செல்லும். இம்மாதிரி சுமார் 5 முறைகள் அச்சிட்டால் ஓர் ஆங்கில எழுத்து உருவாகும். (படம் 2.12).



படம் 2 - 11 அச்சிடும் தலை



படம் 2 - 12 புள்ளி மூலம் ஓர் எழுத்து

ஒவ்வொரு புள்ளியாக அச்சிட முடிவதால், நமக்குத் தேவையான எந்த மொழியிலும் அச்சிடலாம். படங்களையும் அச்சிடலாம். தனித்தனிப் புள்ளிகளாக அச்சிடுவதால் எழுத்துகள் தரமாக இருக்காது. புள்ளிகளை நெருக்கமாக அடித்து

எழுத்துக்களை ஓரளவு அழகாகச் செய்யலாம். இதற்குச் சற்று அதிக நேரமாகும். இதை near letter quality - NLQ என்பர்.

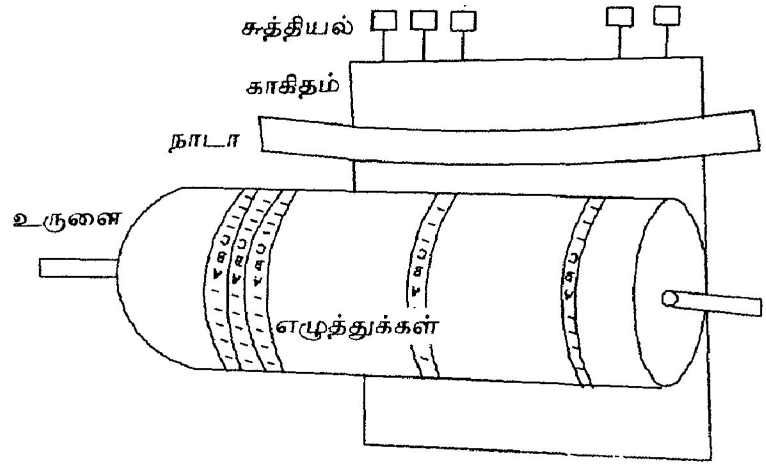
24 புள்ளி வகையில் புள்ளிகளின் அளவு சிறியதாக இருக்கும். அதனால், எழுத்துக்கள் தரமாக இருக்கும்.

சாதாரணமாகப் புள்ளிவகை அச்சுப் பொறிகளில் ஒரு வண்ணத்தில்தான் அச்சிட முடியும். ஆனால் இப்போது வரும் பொறிகளில் சில மாற்றங்களைச் செய்து பல வண்ணங்களில் அச்சிட வகை செய்திருக்கிறார்கள். ஒரே சமயத்தில் எழுத்துகளையும் படங்களையும் பல வண்ணங்களில் அழகாக அச்சிட முடியும்.

வரி அச்சுப்பொறி (Line Printer)

நாம் பயன்படுத்தும் சாதாரண தட்டச்சுப் பொறியில் காகிதமும், ஒரு நாடாவும், ஓர் உருளையின் மீது படிந்திருக்கும். ஓர் எழுத்து உள்ள சுத்தியல் போன்ற ஒரு சிறு தலை நாடாவையும் காகிதத்தையும் தட்டும். அப்போது அந்த எழுத்து காகிதத்தில் பதிவாகும். இதையே சற்று மாற்றிப் பார்ப்போம். உருளையின் மேல் எழுத்துகள் இருக்கட்டும். சுத்தியல் சாதாரணமாக இருக்கட்டும். இப்போதும் சுத்தியல் தட்டினால் காகிதத்தில் ஒரு எழுத்து பதிவாகும். இதற்கு நாடாவானது உருளைக்கும் காகிதத்திற்கும் நடுவில் இருக்க வேண்டும்.

இந்த அமைப்பில், உருளை தன் அச்சில் சுழல முடியும்; வேறெந்த விதத்திலும் நகரமுடியாது. அதனால் தேவையான எழுத்துகள் எல்லாம் ஒரு வளையம் போல் உருளையின்மேல் ஒவ்வொன்றாகச் செதுக்கப்பட்டிருக்கும்; ஒரு நாணயத்தின் வெளி வரம்பில் உள்ள கோடுகள் போல். உருளை சுழன்று கொண்டே இருக்கும்போது தேவையான எழுத்து சுத்தியலின் கீழ் வரும்போது சுத்தியல் செயல்பட்டு அந்த எழுத்தைத் தட்டும். ஒரு வரியில் 132 எழுத்துகள் தேவையென்றால், 132 எழுத்து வளையங்களும், 132 சிறு சுத்தியல்களும் செயல்படும் (படம் 2.13). ஒவ்வொரு சுத்தியலும் தனித்தனியாக, சுதந்திரமாகச் செயல்படும். இதனால் உருளை ஒரு சுற்று சுற்றி முடிப்பதற்குள்ளேயே ஒரு வரியில் உள்ள எல்லா எழுத்துகளும் அச்சிடப்பட்டுவிடும். பிறகு காகிதம் நகர்ந்து அடுத்த வரிக்குச் செல்லும். இப்படி ஒரு சமயத்தில் ஒரு வரியை முழுவதும் அச்சிடுவதால் இவ்வகைப் பொறிகளுக்கு வரி அச்சுப்பொறிகள் என்று பெயர். இவற்றின் வேகம் அதிகம். நிமிடத்திற்கு 200 முதல் 3000 வரிகள் வரை அச்சிடும் பொறிகள் உள்ளன. வேலைப்பளு அதிகம் உள்ள பெரிய கணிப்பொறி மையங்களில் இவற்றைக் காணலாம்.



படம் 2 - 13 வரி அச்சுப்பொறி

லேசர் அச்சுப்பொறி (Laser Printer)

லேசர் என்னும் ஒளிக்கற்றையைப் பயன்படுத்தி மிகச்சிறு புள்ளிகளை உருவாக்க முடியும். ஓர் அங்குலத்தில் 150 அல்லது 300 அல்லது அதற்கு மேலும் புள்ளிகளை இடமுடியும். அதாவது ஒரு சதுர அங்குலத்தில் 22,500 அல்லது 90,000 அல்லது அதற்கும் மேல். இப்புள்ளிகளின் அளவு மிகவும் சிறியதாகையால், கிடைக்கும் எழுத்துகளின் தரம் அதிகம். நிமிடத்திற்கு சில பக்கங்களே அச்சிட முடியும்.

கணிப்பொறியின் திறமையினால் எழுத்துகளை பல அளவுகளிலும், பல வடிவங்களிலும் எழுத முடியும். இப்படி அழகிய எழுத்துகளும் படங்களும் அடங்கிய பக்கங்களை சில நிமிடங்களில், நம் ரசனைக்கேற்ப மாற்றி அமைத்துக் கொள்ளலாம். இவ்விதத்தில் கணிப்பொறி வடிவமைத்து லேசர் பொறியில் அச்சிடப்பட்ட பக்கங்களைப் படமெடுத்து ஆஃப்செட் (Offset Printing) முறையில் அச்சிடலாம். இப்புத்தகம் கூட அவ்வாறுதான் வெளிவந்துள்ளது.

மேஜைக் கணிப்பொறி, தரமான லேசர் அச்சுப்பொறி, பக்கங்களை வடிவமைக்கும் மென்பொருள் இவற்றைக் கொண்டு அச்சிடும் முறை **மேஜைப் பதிப்பு** (Desk Top Publishing) எனப்படும். ஆப்பிளின் வழித் தோன்றலான மக்கின்டாஷ் என்னும் கணிப்பொறிகள் மேஜைப் பதிப்பில் சிறப்பான ஒரு தனி இடத்தைப் பெற்றுள்ளன.

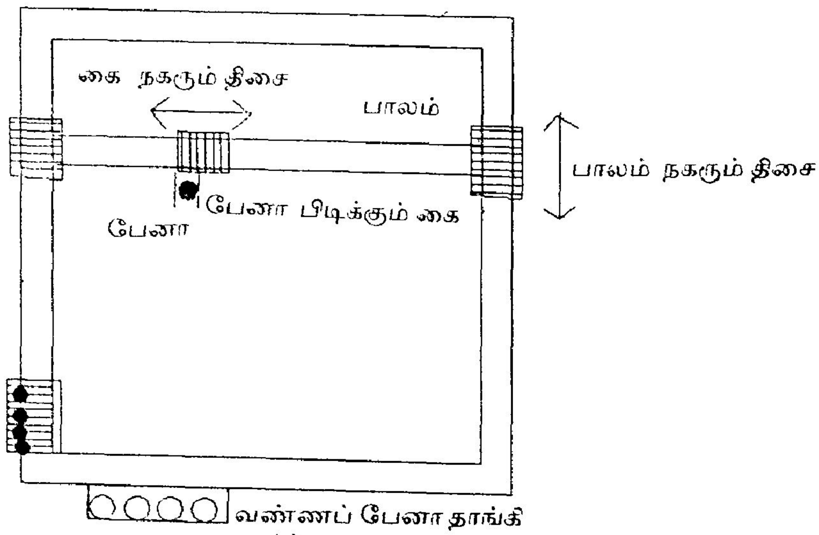
பத்திரிகைகளும், செய்தித் தாள்களும் இன்று பெரும்பாலும் கணிப்பொறி உதவியுடனேயே வெளிவருகின்றன.

மை பீச்சு அச்சுப்பொறிகள் (Ink Jet Printer) இப்போது கிடைக்கும் புதுவகை அச்சுப்பொறிகள். இவை லேசர் பொறியின் தரத்தை எட்ட முனைகின்றன. ஆனால்

விலை குறைவு. பல வண்ணங்களிலும் அச்சிடலாம். லேசர் அச்சுப்பொறி போன்று சிறிய புள்ளிகளை உருவாக்கி எழுதும்/வரையும்.

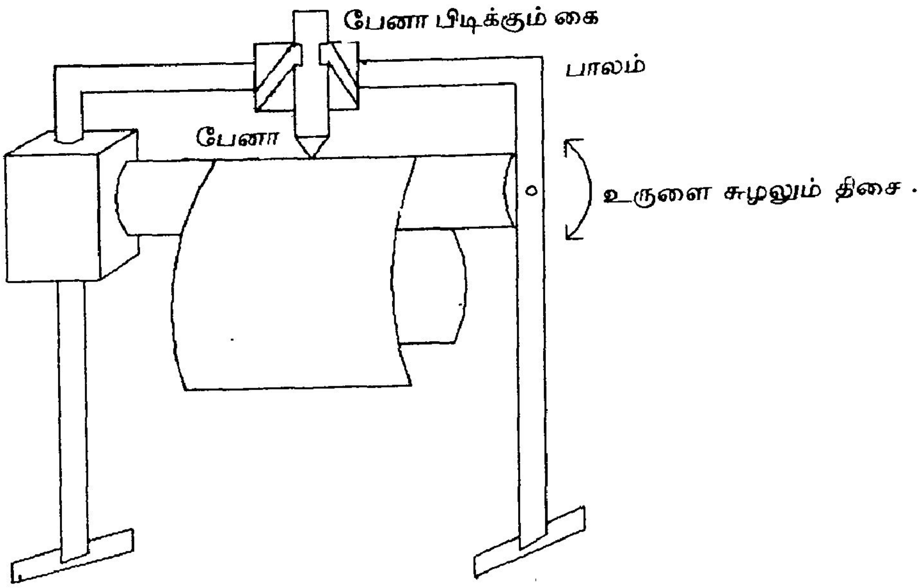
வரைவான் (Plotter)

வடிவமைப்புப் படங்களை (Design Drawings) பெரிய அளவு வரை படத்தாள்களில் வரையப் பயன்படும் சாதனங்கள் இந்த **பேனா வரைவான்கள்** (Pen plotters). இவற்றில் இரண்டு வகைகள் உண்டு. **சமதள வரைவான்** (Flat bed plotter) எனப்படும் வகையில் *(படம் 2.14)* குறிப்பிட்ட அளவுள்ள வரைபடத்தாள் ஒரு சமதளத்தின் மேல் அசையாமல் பொருத்தப்படும். பேனாவைப் பற்றிக்கொள்ள ஒரு சிறு கை இருக்கும். இந்தக் கை கீழே நகர்ந்தால் பேனா காகிதத்தைத் தொடும்; மேலே நகர்ந்தால் பேனா காகிதத்தைத் தொடாது. பேனா பிடிக்கும் கை ஒரு பாலத்தின் மீது ஒரு கோடியிலிருந்து மற்றொரு கோடி வரை நகர முடியும். இந்தப் பாலம் முழுவதுமே காகிதத்தின் ஒரு பக்கத்திலிருந்து எதிர்ப்பக்கம் வரை நகர முடியும். இந்த இரண்டு வகை அசைவுகளையும் பயன்படுத்தி, காகிதத்தின் எந்த இடத்திற்கும் பேனாவைக் கொண்டு செல்லலாம். இப்படிச் செல்லும்போது வரைந்துகொண்டே செல்லலாம் அல்லது வரையாமலும் செல்லலாம்.



படம் 2 - 14 சமதள வரைவான்

சிலமாதிரிப் பொறிகளில் ஒரே ஒரு பேனா மட்டும் இருக்கும். வேறு சில மாதிரிகளில், ஒரு பக்கத்தில் வண்ணப் பேனாக்கள் பல ஒரு தாங்கியில் வரிசையாக அடுக்கி வைக்கப்பட்டிருக்கும். கருவியின் கை தானாக வந்து தேவையான வண்ணப் பேனாவை எடுத்துச் சென்று வரையும். அடுத்த வண்ணம் தேவையானபோது, கையிலுள்ள பேனாவை அதற்கான இடத்தில் வைத்துவிட்டு, அடுத்த பேனாவைப் பற்றி எடுக்கும்.



படம் 2 - 15 நிற்கும் வகை வரைவான்

இரண்டாம் வகைப்பொறி நிற்கும் நிலையில் இருக்கும் (படம் 2.15). இதன் மேல்முனையில் உள்ள ஓர் உருளையின் மீது வரைபடத்தாள் படிந்து இருக்கும். உருளை நகரும்போது காகிதம் முன்னும் பின்னும் நகரும். உருளைக்கு மேலே உள்ள ஒரு பாலத்தில் பொருத்தப்பட்ட ஒரு கை, பாலத்தின் ஒரு கோடியிலிருந்து மறுகோடி வரை நகரும். காகிதத்தையும், கையையும் நகர்த்தி பேனாவை காகிதத்தின் எந்தப் புள்ளிக்கும் கொண்டு செல்லலாம். இப்பொறியில் பயன்படும் காகிதத்தின் அகலம், பொறியின் அகலத்திற்கு உட்பட்டு இருக்க வேண்டும். ஆனால் நீளம் எவ்வளவு வேண்டுமானாலும் இருக்கலாம். வரைவான்கள் வரையும் படங்கள் மிகத் துல்லியமாக இருக்கும். இவை படங்களை விரைவாக வரையும். படங்களில் தேவையான மாறுதல்கள் செய்தாலும் உடனுக்குடன் விரைந்து படங்களைப் பெறமுடியும்.

நினைவில் நிற்க

1. நமக்குப் படிக்க முடியும் வகையில் அச்சடித்துத் தரும் சாதனங்களில் முக்கியமானவை புள்ளிவரிசை அச்சுப்பொறி, வரி அச்சுப்பொறி, லேசர் அச்சுப்பொறி மற்றும் வரைவான்கள்.

2. புள்ளிவரிசை மற்றும் லேசர் அச்சுப்பொறிகள், சிறு புள்ளிகள் மூலம் எழுத்துக்களையும், படங்களையும் உருவாக்கும்.

3. வரி அச்சுப்பொறியில் எழுத்துகள் ஒவ்வொன்றும் சுத்தியலின் ஒரு தட்டிலேயே முழுமையாக உருவாகும். வரிவரியாக எழுத்துகள் உருவாவதால் இதன் வேகம் அதிகம். ஆனால் இது படங்கள் வரையப் பயன்படாது.

4. லேசர் அச்சுப்பொறிகள் உருவாக்கும் புள்ளிகள் மிகச் சிறியவை; அதனால் அச்சுத்திறன் மிக உயர்ந்தது. மேஜைப் பதிப்பிற்குப் பயன்படுகின்றன.

5. வடிவமைப்புப் படங்களை வரைவதற்குப் பயன்படுவது பேனா வரைவான்கள். இவை ஒன்று அல்லது அதற்கு மேற்பட்ட வண்ணங்களில் வரையும். துல்லியமாக வரையும் திறன் பெற்றவை. விரைவாகவும் வரையும்.

பயிற்சி வினாக்கள்

1. கீழ் உள்ள அச்சுப்பொறிகளின் அமைப்பையும், செயல்படும் விதத்தையும் சுருக்கமாக விவரி.

   அ. புள்ளிவரிசை அச்சுப்பொறி　　ஆ. வரி அச்சுப்பொறி
   இ. லேசர் அச்சுப்பொறி

2. புள்ளிவரிசை அச்சுப்பொறிக்கும், வரி அச்சுப்பொறிக்கும் உள்ள வேற்றுமைகள் மூன்றினைக் கூறுக.

3. மேஜைப் பதிப்பு என்றால் என்ன? இதற்குப் பயன்படும் அச்சுப்பொறி எது?

4. வரைவான்களின் பயன்களைக் கூறுக.

5. வரைவான்கள் செயல்படும் விதத்தை விவரி.

2.7 கணிப்பொறி செயல்படும் விதங்கள்

காலம் மாற மாற கணிப்பொறிகள் செயல்படும் விதத்திலும் மாற்றங்கள் ஏற்பட்டுள்ளன. அவற்றைப் பற்றி இப்போது தெரிந்து கொள்வோம்.

நேரடி முறை

ஆரம்ப காலத்தில் கட்டளை நிரல் எழுதியவர்களே கணிப்பொறியை இயக்கிக் கணக்கிட்டனர். ஒருவர் மாற்றி ஒருவர் இயக்கும்போது கால விரயம் அதிகமானது. உதாரணமாக, இருவர் வருவதாக வைத்துக்கொள்வோம். முதலில் வந்தவர் ஃபோர்ட்ரான் மொழி மாற்றியைக் கணிப்பொறிக்குள் செலுத்துகிறார். நினைவிருக்கட்டும் - அன்று இந்த மொழிமாற்றி துளை அட்டைகளில் பதிவு செய்யப்பட்டிருந்தது. துளை அட்டை படிப்பான் மூலம் இந்த மொழிமாற்றி கணிப்பொறிக்குள் செல்கிறது. அடுத்தது கட்டளை நிரல் இதே போல் உள்ளே செலுத்தப்படுகிறது. மொழிமாற்றி இந்தக் கட்டளை நிரலை பொறிமொழியில் மாற்றி, அதைத் துளை அட்டைகளில் பதிவு செய்து கொடுக்கிறது. இந்த அட்டைகளை மீண்டும் உள்ளே செலுத்தி, தேவையான கணிப்பைச் செய்ய வேண்டும். அவர் சென்றவுடன், அடுத்தவரும் இதே வேலைகளைத் திரும்பவும் செய்வார். இப்படிப் பலரும் ஃபோர்ட்ரான் மொழிமாற்றியைப் பலமுறை உள்ளீடு செய்ய வேண்டியிருந்தது. இதனால் காலதாமதம் அதிகமானது.

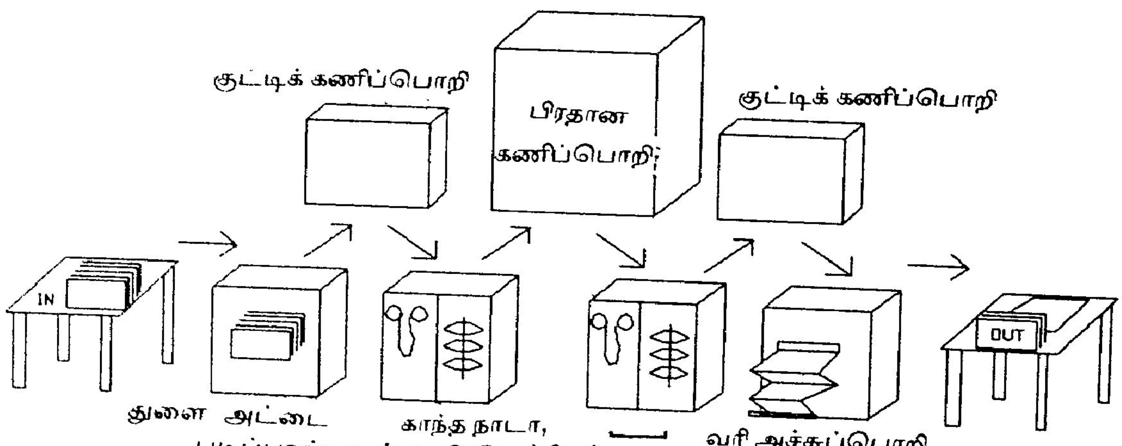
தொகுப்பு முறை (Batch Processing)

ஒருமுறை ஃபோர்ட்ரான் மொழிமாற்றியை உள்ளீடு செய்தால், வரிசையாகப் பல கட்டளை நிரல்களை மொழிமாற்றம் செய்து கொண்டால் கணிப்பொறிக்கு நிறைய நேரம் மிச்சமாகும். அந்தக் காலத்தில் கணிப்பொறிகளின் விலை மிக அதிகமாக இருந்தது. இதனுடன் ஒப்பிடும்போது **நிரலர்களின்** (Programmers) சம்பளம் மிகவும் குறைவாக இருந்தது. அதனால், நிரலரின் நேரம் வீணானால் பரவாயில்லை, கணிப்பொறியின் நேரம் வீணாகக் கூடாது என்று கருதி **தொகுப்பு முறையைக்** கொண்டு வந்தார்கள்.

இம்முறையில், எல்லோரும் கட்டளை நிரல்களை துளை அட்டைகளில் பதிவுசெய்து, கணிப்பொறியை நிர்வகிப்பவர்களிடம் கொடுத்துவிட வேண்டும். ஒரு குறிப்பிட்ட நேரத்தில், தேவையான மொழி மாற்றியைப் பொறுத்து, இந்த நிரல்கள் எல்லாம் பல தொகுதிகளாகப் பிரிக்கப்படும். ஒவ்வொரு தொகுதியாக, துளை அட்டை படிக்கும் பொறியில் இடப்படும். இவை படிக்கப்பட்டு, கணிப்பொறி விரைவில் படிக்க ஏதுவாக, காந்த நாடாக்கள் அல்லது வன்தகடுகளில் பதிவு செய்யப்படும். கணிப்பொறியானது, ஒருமுறை உள்ளீடு செய்யப்பட்ட ஒரு

மொழிமாற்றியைக் கொண்டு, எல்லா நிரல்களையும் வரிசையாக மொழிமாற்றம் செய்யும். நிரல்களில் தவறிருந்தால் அவற்றைச் சுட்டிக்காட்டும். தவறில்லாத நிரல்களை, தேவையானால் செயல்படுத்தி அவற்றுக்கான விடைகளைக் கண்டு பிடிக்கும். இப்படி, தவறுகளின் பட்டியல், அல்லது விடைகள் போன்றவற்றை, கணிப்பொறியானது மீண்டும் நாடாக்கள் அல்லது தகடுகளில் பதிவு செய்து வைக்கும். இதிலிருந்து அச்சுப் பொறிகள் மூலம் வெளியீடு செய்து, வரிசையாக வரும் வெளியீடுகளை அதனதன் நிரல்களுடன் இணைத்து, குறிப்பிட்ட நேரத்தில் திரும்பக் கொடுப்பார்கள்.

ஒருநாளைக்கு, சாதாரணமாக இரண்டு மூன்று முறை இப்படி நடக்கும். ஒரு தவறு இருந்தாலும் நிரலருக்கு அரைநாள் வீணாகும். ஆனால் கணிப்பொறியின் வேகம் இருவழிகளில் அதிகமாகியது. முதல் வழி - ஒருமுறை உள்ளீடு செய்த மொழிமாற்றியைப் பலமுறை பயன்படுத்துவது. அடுத்த வழி - மிக மெதுவாகச் செயல்படும் துளை அட்டை படிப்பானுக்காகவும், அச்சுப்பொறிக்காகவும் கணிப் பொறி காத்திருப்பதில்லை. விரைவாகச் செயல்படும் வன்தகட்டிலிருந்து, அல்லது காந்த நாடாவிலிருந்து தகவலை எடுத்து, திரும்பவும் அதிலேயே தகவலை வைக்கும். துளை அட்டைகளைப் படிப்பதையும், அச்சடித்துக் கொடுப்பதையும் வேறு குட்டிக் கணிப்பொறிகள் பார்த்துக் கொள்ளும். IBM கணிப்பொறி இவ்வகையில் செயல்பட்டது. இந்த அமைப்பைப் படம் 2.16 விளக்குகிறது.



படம் 2 - 16 தொகுப்பு முறை

மீண்டும் நேரடி முறை

சில வருடங்களில் கணிப்பொறியின் விலை குறைந்தது. நிரலர்களின் சம்பளம் கூடிக்கொண்டே போனது. அதனால் நிரலர்களின் நேரம் வீணாவதைத் தவிர்க்க வேண்டியது அவசியமானது. இதற்காக நிரலர் கணிப்பொறியுடன் நேரடியாகத்

தொடர்பு கொள்ளும் முறை இப்போது மீண்டும் வந்துவிட்டது. கணிப்பொறி இயலில் ஏற்பட்டுள்ள முன்னேற்றம் இதற்கு வழி செய்து கொடுத் துள்ளது.

நுண்கணிப்பொறிகள் ஒருவர் பயன்படுத்துவதற்காகவே வடிவமைக்கப் பட்டன.

பெரிய கணிப்பொறிகளில் பல **முனையங்கள்** (Terminals) இணைக்கப் படுகின்றன. ஒவ்வொன்றிலும் ஒருவர் உட்கார்ந்து கணிப்பொறியுடன் நேரடியாக உரையாடி தேவையானவற்றைக் கணித்துக் கொள்ளலாம். இந்த முறையில், கணிப்பொறியானது ஒவ்வொரு வினாடியிலும் ஒரு சிறு பாகத்தை, அதாவது சில மில்லி வினாடிகளை, வரிசையாக ஒவ்வொருவருக்கும் செலவிடும். அப்போது அவர்களுக்கான வேலைகளைச் செய்து கொடுக்கும். கணிப்பொறியின் வேகம் அதிகமாதலால், அது தன்னுடன் மட்டும் உரையாடுகிறது என்ற தோற்றத்தை ஒவ்வொருவர் மனத்திலும் ஏற்படுத்தும். கண்ணன் தன்னுடன் மட்டுமே ஆடிப்பாடுவதாக ஒவ்வொரு கோபிகையும் நினைப்பாளாம். அதே போலத்தான் இதுவும். இது **நேரப்பகிர்வு முறை** (Time sharing) எனப்படும். ஒவ்வொரு முனையத்தில் இருப்பவரும் அதிக வேலையை ஒரே சமயத்தில் கொடுத்தால், வேலை தாமதமாவது ஒவ்வொருவருக்கும் நன்றாகத் தெரியும்.



பலபணிமுறை (Multitasking)

ஒரு நிரலில் (அல்லது பல நிரல்களில்) செய்ய வேண்டிய வேலைகளை பல சிறு பணிகளாகப் பிரிக்கலாம். இவற்றில் சில, உள்ளீடு செய்ய வேண்டிய தகவல்களைப் படிப்பது, நினைவகத்திலிருந்து தகவல்களைப் பெறுவது, தகவல்களை நினைவகத்திற்கு அல்லது அச்சுப்பொறிக்கு அனுப்புவது, சில் கணிதச் செயல்பாடுகளைச் செய்வது. கணிதச் செயல்பாடுகளுடன் ஒப்பிடும்போது நினைவகம் மற்றும் உள்/வெளியீட்டகங்களைப் பயன்படுத்தும் செயல்பாடு களுக்குச் சற்று அதிக நேரமாகும். உள்ளீடு அல்லது வெளியீடு செய்யும்போது, அந்தப்பணி முடியும் வரை மையச் செயலகம் காத்துக்கொண்டிருக்கும். பிறகு அடுத்த பணியைச் செய்யும். சில இயக்க அமைப்புகள் இம்மாதிரி மையச்செயலகம் நேரத்தை வீணாக்குவதைத் தவிர்த்துவிடும். மையச் செயலகம் காத்திருக்கும் நேரத்தில், கணிதச் செயல்பாடு போன்றவை ஏதேனும் செய்யவேண்டியிருந்தால், அதைக் கொடுத்து செய்யச் சொல்லும். இப்படி பல பணிகளை மாற்றி மாற்றிச் செய்வது கணிக்கும் நேரத்தைக் குறைக்கும். இம்முறை **பலபணி முறை** எனப்படும்.

ஒரு பணியை, பாதியில் விட்டுச் செல்லும்போது, ஒவ்வொரு முறையும், அந்தப் பணியில் எவ்வளவு முடிந்திருக்கிறது என்ற விவரங்களுடன், அந்தப் பணியைப் பாதுகாப்பாகப் பத்திரப்படுத்தி வைக்க வேண்டும்.

உடனடி முறை (On-Line Processing)

நுண்கணிப் பொறிகளை இயக்கும்போதும், பெரிய கணிப்பொறிகளில் ஒரு முனையத்தின் மூலம் தொடர்பு கொள்ளும்போதும் நமது கட்டளைகள் உடனுக் குடன் செயலாக்கப்படுகின்றன. இது உடனடி முறை எனப்படும். இம்முறையில் நாம் பயன்படுத்தும் எல்லா சாதனங்களையும் உடனடியாகப் பயன்படுத்த முடிகிறது.

இரயில் வண்டிகளுக்கு பயணச்சீட்டு கொடுக்கும் கணிப்பொறிக்குப் பல முனையங்கள் உண்டு. எல்லா முனையங்களும் ஒரே கட்டளை நிரலின் இயக்கத்தில் தான் பணி செய்கின்றன. எல்லாருடைய கட்டளைகளும் உடனுக்குடன் நிறை வேற்றப்படுகின்றன. இது உடனடி முறைக்கு ஓர் உதாரணம்.

தொகுப்பு முறையில் நாம் கட்டளை நிரல்களைக் கொடுக்கும் நேரம் ஒன்று. கணிப்பொறி மையத்தின் சௌகரியத்திற்கு ஏற்றாற்போல் ஒரு சமயத்தில் சில செயல்பாடுகள் நடக்கும். வேறு சமயத்தில் வேறு சில செயல்பாடுகள் (உதாரண மாக, அச்சிடுவது) நடக்கும். இவ்வாறு செயல்படும் விதம் **உடனடியற்ற முறை** (Offline processing) எனப்படும்.

நிகழ்காலச் செயல்பாட்டு முறை (Real time processing)

பெரும்பாலான நிரல்கள் செயல்படும்போது, வரும் விடைகள்தான் முக்கியமே தவிர, அதற்கு எவ்வளவு நேரமாகிறது என்பது முக்கியமில்லை. உதாரணமாக, ஒரு சம்பளப் பட்டியல் தயாரிக்க ஒரு மணிநேரம் ஆனதா அல்லது 2 மணி நேரம் ஆனதா என்பது மிக மிக முக்கியமான விஷயம் இல்லை. ஆனால், சில சமயங்களில் கணிப்பொறி எவ்வளவு விரைவாக விடை கொடுக்கிறது என்பது மிக முக்கியம். உதார்ணமாக, ஓர் அணு உலையை அல்லது இரசாயனத் தொழிற் சாலையை அல்லது ஒரு விமானத்தை இயக்கும் கணிப்பொறியை எடுத்துக் கொள்ளலாம். இங்கெல்லாம், நிகழும் மாற்றங்களை உணர்ந்து, அவற்றுக்கேற்றாற் போல் தேவையான கணிப்புகளைச் செய்து, தேவையான மாற்றங்களைச் செய்ய உடனுக்குடன் ஆணைகள் பிறப்பிக்க வேண்டும். இந்த **உடனுக்குடன்** என்பது ஒன்றிரண்டு மில்லி வினாடிகள் அல்லது சில வினாடிகள் என்று பலவிதமாகப் பொருள்படும் - சூழ்நிலையைப் பொறுத்து. இவ்வளவு குறுகிய காலத்திற்குள் கணிப்பொறியானது விடையைக் கணித்து,. சூழ்நிலையைக் கட்டுப்படுத்த வேண்டும். இவ்வாறு நிகழ்கால நிகழ்ச்சிகளைக் கட்டுப்படுத்தும் முறை **நிகழ்காலச் செயல்பாட்டு முறை** எனப்படும். இரயில் நிலையங்களில் பயணச் சீட்டு பதிவு செய்வது உடனடி முறை; ஆனால் நிகழ்காலச் செயல்பாட்டு முறை இல்லை.

மோடம் (Modem)

பெரிய கணிப்பொறி ஒன்றில் வேலை செய்ய அது இருக்குமிடத்திற்குத்தான் போகவேண்டும் என்பதில்லை. நாம் இருக்குமிடத்திலேயே உள்ள ஒரு நுண்கணிப் பொறியைக் கொண்டு, தொலைபேசி இணைப்பு மூலம் பிரதான கணிப்பொறி யுடன் தொடர்பு கொண்டு, நம் கணிப்பொறியை பிரதான கணிப்பொறியின் ஒரு முனையமாக மாற்றிக் கொள்ளலாம். இப்படிச் செய்யும்போது பிரதான கணிப்பொறிக்கும் அதன் முனையத்திற்கும் இடையில் செல்லும் மின் சமிக்ஞைகள் தொலைபேசித் தொடர்பிற்கு ஏற்ற வகையில் இருக்கவேண்டும். ஆனால், கணிப்பொறியில் செயல்படும் சமிக்ஞைகள் வேறுவிதமாக இருக்கும். அதனால், கணிப்பொறியையும் தொலைபேசித் தொடர்பையும் இணைக்கும் இடத்திலும், சமிக்ஞை மாற்றிகள் தேவை. இந்த மாற்றிகள், கணிப்பொறி சமிக்ஞைகளைத் தொலைபேசிச் சமிக்ஞைகளாகவும், தொலைபேசிச் சமிக்ஞைகளை கணிப்பொறிச் சமிக்ஞைகளாகவும் மாற்றும் திறன் பெற்றவை. அதனால் இவை மோடம் (Modem - Modulator Demodulator) எனப்படும்.

பின்னலமைப்பு (Network)

ஒரு தொழிற்சாலையில், பிரதான கணிப்பொறி மையம் ஒன்றை அமைத்து, எல்லாக் கணிப்புகளையும் ஒரே இடத்தில் செய்வது பழைய முறை. பல நுண்கணிப்பொறிகள் இணைந்த ஒரு **பின்னலமைப்பு** மூலம் செயல்படுவது புதியமுறை. உதாரணமாக, பொருள்கள் உள்ளே வரும் இடத்தில் ஒரு கணிப்பொறி இருந்தால், வரும் பொருள்கள் பற்றிய விபரங்களை உடனுக்குடன் உள்ளீடு செய்யலாம். இதேபோல் பொருள்களைப் பல துறைகளுக்குப் பகிர்ந்தளிக்கும் இடத்திலும், கணக்குப் பார்த்து பொருள்களுக்குப் பணம் தரும் இடத்திலும், விற்பனைப் பிரிவிலும் கணிப்பொறிகள் செயல்பட்டால் வேலை சுலபமாகும். ஆனால் அவை தனித்தனியாகச் செயல்பட்டால் பயனில்லை. உள்ளே வரும் பொருள்களின் விபரம் பணம் தருபவருக்கு உடனடியாகத் தெரிந்தாக வேண்டும். இதற்குக் கணிப்பொறிகளிடையே இணைப்பும், அதன்மூலம் தகவல் பரிமாற்றமும் அவசியம். கணிப்பொறிகள் எல்லாம் ஒன்றிரண்டு கிலோ மீட்டர் தூரத்திற்குள் இருந்தால் அவற்றைக் கம்பிகள் மூலம் இணைத்து, ஒரு **சிறுபரப்புப் பின்னலமைப்பை** (LAN - Local Area Network) உருவாக்கலாம். இந்தப் பின்ன லமைப்பை நிருவகிக்க ஒரு **பின்னலமைப்பு இயக்க முறை** (Network Operating System) தேவை.

ஒரு நிறுவனத்தின் கிளைகள் உலகத்தின் பல பாகங்களிலும் இருந்தால், அவற்றிலெல்லாம் உள்ள கணிப்பொறிகளை இணைத்து உருவாக்கப்படுவது **பெரும்பரப்புப் பின்னலமைப்பு** (WAN - Wide Area Network) எனப்படும். இங்கு கணிப்பொறிகளுக்கிடையே இணைப்பை ஏற்படுத்த தொலைபேசி, மைக்ரோ அலை, செயற்கைக் கோள் முதலியவை பயன்படும். கணிப்பொறிப் பின்ன லமைப்பைப் பயன்படுத்துவதில் பல நன்மைகள் உண்டு. தகவல் எங்கிருந்தாலும் தேவைப்பட்ட இடங்களில் விரைந்து பெறலாம். சிறிய கணிப்பொறிகளைப் பயன் படுத்துவதால் செலவும் குறைவு.

நுண்கணிப்பொறிகளின் வகைகள்

இவை தோன்றிய காலத்தில் **சொந்தக் கணிப்பொறி** (PC - Personal Computer), **மேஜைக் கணிப்பொறி** (Table top computer) என்று அழைக்கப் பட்டன. இந்த வகைப் பொறிகளில் பெரும்பாலும் முதலில் இன்டெல் நிறுவனம் தயாரித்த 8088 என்ற மையச்செயலகச் சில்லு பயன்பட்டது. இவற்றின் திறன் குறைவு, வேகமும் குறைவு. விலையைக் குறைப்பதற்காக நெகிழ்தகடுகள் பயன்படுத்தப்பட்டன. வன்தகடுகள் இல்லை.

சற்று அதிகத் திறன் படைத்ததாக, வன்தகடுகளைப் பயன்படுத்துவதாக அமைந்த அடுத்த கட்ட நுண்கணிப் பொறிகள் PC/XT (XT - Extended Technology)

எனப்படும். தொழில்நுட்பம் முன்னேற முன்னேற இன்டெல் நிறுவனம் 80286, 80386, 80486, பென்டியம் என்ற புதிய சில்லுகளை அறிமுகப்படுத்தியது. இவற்றைக் கொண்டு உருவாக்கப்பட்ட கணிப்பொறிகள் PC/AT (AT - Advanced Technology) *எனப்பட்டன.* 640 KB*க்கும் அதிகமான நினைவகம், அதிக அடர்வு நெகிழ்தகடு, பலபணித்திறம், அதிக வேகம் என்பவை சில முக்கிய முன்னேற்றங்கள்.*

PC, PC/XT, PC/AT *என்று வித்தியாசப்படுத்திப் பார்க்கும்போது, அப்பொறியில் பயன்படும் மையச்செயலகச் சில்லு, தாய்ப்பலகையில் செல்லும் பஸ்களின் அளவு, தாய்ப்பல்கையின் தொழில்நுட்பம், உள்/வெளித் தொடர்பு போன்ற பல விஷயங்களைப் பார்க்க வேண்டும்.* PS/2 (Personal System 2) *என்பது* IBM *வெளியிடும் புதிய வகை நுண்கணிப்பொறிகள். ஆப்பிள், மக்கின்டாஷ் என்பவை ஒரு தனி இன நுண்கணிப்பொறிகள்.*

செயற்கை அறிவு (Artificial Intelligence)

ஐந்தாம் தலைமுறைக் கணிப்பொறிகள் செயற்கை அறிவு பெற்றதாக இருக்க வேண்டும் என வடிவமைக்கப்படுகின்றன என்று ஏற்கனவே பார்த்தோம். பல எண்களைப் பெருக்குவது, தேவையான கணக்குகளைப் போட்டு முடிப்பது போன்றவை, கணிப்பொறிகளிடம் நாம் சாதாரணமாக எதிர்பார்ப்பது. இவற்றைச் செய்வதனால் கணிப்பொறி அறிவுபடைத்தது என்று சொல்ல முடியாது. பல்லாயிரக்கணக்கான கணித தர்க்கச் செயல்பாடுகளை, சில வினாடிகளில் செய்து முடிப்பது அறிவுத்திறன் ஆகாது. படத்தைப் பார்த்து அதில் உள்ள பொருள்களையும், மனிதர்களையும் அடையாளம் காண்பது, வாக்கியங்களைப் படித்து அவற்றின் பொருளைப் புரிந்துகொள்வது போன்றவை சாதாரணக் கணிப்பில் அடங்காது. இவற்றைச் செய்ய அறிவுத்திறன் வேண்டும். இவற்றையும் செய்யும் அளவுக்கு மிக அருமையாக எழுதப்பட்ட மென்பொருள் கணிப்பொறியில் செயல்படும்போது, 'கணிப்பொறிக்கு இவ்வளவு அறிவா!' - என்று வியக்குமளவுக்கு செயல்படும். இதுதான் செயற்கை அறிவு எனப்படும்.

நினைவில் நிற்க

1. பல கட்டளை நிரல்களைச் சேர்த்து, ஒரு தொகுதியாக்கி, அவற்றை ஒரே சமயத்தில் ஒன்றன் பின் ஒன்றாகச் செயல்படுத்துவது தொகுப்பு முறை எனப்படும். மையச்செயலகம் அதிக நேரம் செயலாற்ற முடிந்தது. ஆனால், நிரலரின் நேரம் வீணானது.

2. ஒரு கணிப்பொறி தன் மையச் செயலகத்தின் நேரத்தைப் பலருக்கும் பகிர்ந்துகொடுத்து எல்லோருடைய வேலைகளையும் செய்து கொடுப்பது நேரப்பகிர்வு முறை.

3. பல பணிகளை, ஒன்றுக்கு நடுவில் ஒன்றாகச் செய்வது பலபணிமுறை. உதாரணமாக, வன்தகட்டிலிருந்து தகவல் வந்து சேரக் காத்துக் கொண்டிருக்கும் நேரத்தில், மையச் செயலகம் சில கணிப்புகளைச் செய்யலாம்.

4. உடனுக்குடன் நம் கட்டளைகளைச் செயல்படுத்துவது உடனடி முறை. கட்டளை நிரல்களை வாங்கிவைத்து பிற்பாடு செயல்படுத்தித் தருவது உடனடியற்ற முறை.

5. மிகக்குறைந்த காலத்தில் கணிப்புகள் செய்து, நடந்து கொண்டிருக்கும் நிகழ்ச்சிகளைக் கட்டுப்படுத்துவது நிகழ்காலச் செயல்முறை.

6. தொலைவிலுள்ள கணிப்பொறியுடன் தொடர்பு கொள்ளும்போது, தொலை தொடர்புச் சாதனங்களுக்கு ஏற்ற வகையில் சமிக்ஞைகளை மாற்ற உதவுவது மோடம்.

7. பல கணிப்பொறிகளை இணைத்து, அவற்றுக்குள் தகவல் பரிமாற்றம் நிகழ வகை செய்வது பின்னலமைப்பு. இதில் சிறுபரப்புப் பின்னலமைப்பு, பெரும் பரப்புப் பின்னலமைப்பு என இருவகை உண்டு.

8. நுண்கணிப்பொறிகளில் PC, PC/XT, PC/AT, PS/2 என பல வகைகள் உண்டு.

9. சாதாரணக் கணிப்பாக இல்லாமல், வியக்க வைக்கும் அளவுக்கு, கணிப்பொறிக்கு செயற்கை அறிவுத்திறன் கொடுப்பது மென்பொருள்களே.

பயிற்சி வினாக்கள்

1. தொகுப்பு முறை என்றால் என்ன? அது எந்தக் காரணங்களினால் உருவாக்கப் பட்டது? அதன் குறை, நிறைகளைக் கூறு.
2. நேரடி முறைக் கணிப்பின் பயன்கள் இரண்டினைக் கூறு.
3. நேரப் பகிர்வு முறையை விவரி. எந்த வகைப் பொறிகளில் இது பயனாகிறது?
4. பல பணிமுறை என்றால் என்ன? இம்முறையில் செயல்படும்போது கவனத்தில் கொள்ள வேண்டியவை எவை?
5. நிகழ்காலச் செயல்பாட்டு முறைக்கும், உடனடி முறைக்கும் உள்ள ஒற்றுமை வேற்றுமைகள் என்ன?
6. கணிப்பொறிப் பின்னலமைப்பு என்றால் என்ன? அதன் நன்மைகள் மூன்றினைக் கூறு.
7. கணிப்பொறிப் பின்னலமைப்பில் உள்ள இரண்டு வகைகளை விவரி.
8. மோடம் என்றால் என்ன? அதன் பயன் யாது?
9. நுண்கணிப்பொறிகளில் உள்ள வகைகள் யாவை?

10. PC, PC/XT, PC/AT என்பவற்றின் விரிவாக்கங்கள் என்ன?

11. PC, PC/XT, PC/AT என்பவற்றுக்குள் உள்ள முக்கிய வித்தியாசங்கள் எவை?

12. நுண்கணிப்பொறிகளின் மையச் செயலகச் சில்லுகள் சிலவற்றைக் கூறு.

13. செயற்கை அறிவு என்றால் என்ன? உதாரணங்கள் கொடு.

2.8 இயக்க அமைப்பு

கணிப்பொறிக்கும், நாம் பயன்படுத்தும் மற்ற மின்னணுக் கருவிகளுக்கும் ஒரு பெரிய வேறுபாடு உண்டு. உதாரணமாக தொலைக்காட்சிப் பெட்டியை எடுத்துக் கொள்வோம். இதை இயக்க ஒரு சில பொத்தான்களே உள்ளன. இதனுடன் இணைக்க VCR, VCP போன்ற துணைக் கருவிகள் மிகச் சில மட்டுமே உள்ளன. இவற்றை இயக்க, சில பொத்தான்களைப் பற்றித் தெரிந்துகொண்டால் போதும். ஆனால் கணிப்பொறி அவ்வாறில்லை.

இதற்கு உள்ளீடு, வெளியீடு செய்ய விதவிதமான பொறிகள் உள்ளன. இவை பல விதங்களில், பல அளவுகளில் கிடைக்கின்றன. இவை ஒவ்வொன்றுடனும் தொடர்பு கொள்ள வெவ்வேறான முறைகளை, விதிகளை, கணிப்பொறி கடைபிடிக்க வேண்டும். இந்த விதிமுறைகளையெல்லாம் நாமே படித்துப் புரிந்து கொண்டு அதன்படி செயலாற்ற வேண்டுமானால், அது நமக்கு மிகவும் கடினமான வேலையாகும். இந்த விதிமுறைகளையெல்லாம் நன்றாக அறிந்த ஒருவர் நமக்கு உதவி செய்தால் எவ்வளவு எளிதாக இருக்கும்?

ஓர் உதாரணத்தைப் பார்ப்போம். விடுமுறையின் சில நாள்களை உல்லாசமாகக் கழிக்க ஊட்டி செல்வதாக வைத்துக்கொள்வோம். அங்கு ஒரு விடுதியில் (Hotel) தங்கச் செல்வோம். விடுதியின் மேலாளர் நமக்கு எந்த அறை என்று கூறுகிறார் - காலியாக இருக்கும் அறையாகப் பார்த்து ஒதுக்குகிறார்! நாம் அங்கு இருக்கும் வரை அதே அறைக்கு வேறு எவரையும் அனுப்பாமலும் இருப்பார். நமக்குத் தேவையான உணவு பற்றிக் கேட்டு, சமையற்காரருக்கு உத்தரவிட, அங்குள்ள சமையற்காரர் நமக்கு வேண்டியவற்றைச் சமைத்துத் தருகிறார். நாமும் மகிழ்ச்சியுடன் ஊட்டியின் வனப்பைக் கண்டுகளித்து ஊர் திரும்பலாம். வரும்போது நம் அறை காலியானதை அவர் குறித்துக்கொள்கிறார்.

இதையே சற்று வித்தியாசமாகப் பார்க்கலாம். விடுதி இருக்கிறது. அதில் சமையலறை சாமான்கள் உட்பட தேவையான பொருள்கள் எல்லாமே இருக்கின்றன. நிருவகிக்கும் ஆள்கள் மட்டும் எவரும் இல்லை என்று வைத்துக் கொள்வோம். இது எப்படியிருக்கும்?

முதலில், எந்த அறை காலியாக இருக்கிறதென்று நாமே பார்க்க வேண்டும்; சமையலறையில் உள்ள அடுப்பு எந்த விதம், எப்படி இயக்க வேண்டும் என்பதைத் தெரிந்து கொள்ள வேண்டும்! நாம் சமையல் செய்யும்போது, மற்றொரு விருந்தினர் நடுவில் நுழைந்துவிட்டால் என்ன செய்வதென்று யோசிக்க வேண்டும்.



1. ''நான் பாயசம் வைக்கிறேன். தண்ணீரும் வெல்லமும் போடுகிறேன்.''
2. ''நான் சாம்பார் வைக்கிறேன். புளி போடுகிறேன்.''
3. ''எனக்கு டீ வேணும். டீத்தூள் போடுகிறேன்.''
4. ''இதென்ன சாம்பார் இப்படி இருக்கு?''

அடுப்பிற்கு எரிபொருள் தீர்ந்து போய்விட்டால் என்ன செய்வது?

இப்படி ஒவ்வொன்றாக அடுக்கிக்கொண்டே போகலாம். இப்படி இருந்தால் நம் ஊட்டி அனுபவம் எப்படி இருக்கும்? இதிலிருந்து என்ன தெரிகிறது? பொருட்கள் மட்டும் விடுதியாவதில்லை. அதை நிருவகிக்க, அங்குள்ள பொருட்களை நன்றாக அறிந்த மேலாளர் மற்றும் பணியாளர்கள் மிக அவசியம்.

கணிப்பொறி விஷயத்தில் நிலைமை இன்னும் சிக்கலானது. பற்பல உள்ளீட்டகங்களையும், வெளியீட்டகங்களையும் சரியான முறையில் நிருவகிக்க வேண்டும். அதனுடன், கணிப்பொறிக்கு உள்ளேயே இருக்கும் நினைவகம் மற்றும் மையச் செயலகம் போன்றவற்றை, கேட்பவர்களுக்குச் சரியாகப் பகிர்ந்தளிக்க வேண்டும். குறுக்கீடுகள் (Interrupts) வந்தால் அவற்றைச் சமாளிக்க வேண்டும்.

மொத்தத்தில், கணிப்பொறியின் எல்லா உறுப்புகளையும் நிருவகித்து, நமக்காகக் கணிப்பொறியின் பல பாகங்களையும் திறம்பட இயக்கித்தர ஒருவர் தேவை. இதைச் செய்யும் மென்பொருள் **இயக்க அமைப்பு** (Operating System) எனப்படுகிறது. கணிப்பொறி உடல் என்றால், அதற்கு உயிராக இருப்பது இயக்க அமைப்பு. இயக்க அமைப்பின் உதவி நமக்கு இருப்பதால், கணிப்பொறியின் உள் விவகாரங்களை அறிந்து கொள்வதில் நம் நேரத்தை வீணாக்காமல், நம் காரியத்தில் கண்ணாயிருக்கலாம். நம் மென்பொருள்கள் நன்கு இயங்க, தேவையான எல்லா வசதிகளையும் செய்து கொடுப்பதே இயக்க அமைப்பின் முக்கியபணி.

இப்போதுள்ள நுண்கணிப்பொறிகளில் மிக அதிகமாகப் பயன்படுவது MS DOS (Micro soft Disk Operating System). இது IBM இனத்தைச் சேர்ந்த, ஆனால், மற்ற நிறுவனத்தாரால் தயாரிக்கப்படும் கணிப்பொறிகளை நிருவகிக்கும். IBM பொறிகளை நிருவகிக்கும் இதே இயக்க அமைப்புக்கு PC DOS (Personal Computer Disc Operating System) என்று பெயர். ஒரு சமயத்தில் ஒருவர் மட்டும் பயன்படுத்து வதற்காக இந்தச் சிறிய பொறிகள் உருவாக்கப்பட்டதால், இந்த இயக்க அமைப்பு கள் எளிமையானவை. தகடுகளை நிருவகிப்பதை முதல் நோக்கமாகக் கொண்டு இவை உருவாக்கப்பட்டதால், இவை DOS என்று பெயர்பெற்றன.

நுண்கணிப்பொறிக்கு மின் இணைப்புக் கொடுத்தவுடன், அது தானாகத் தேடி, முதலில் உள்ளே (நினைவகத்தில்) எடுத்து வைத்துக் கொள்வது இந்த இயக்க அமைப்பின் ஒரு சிறுபகுதியைத்தான். உடனே கணிப்பொறியின் கட்டுப்பாட் டினை இந்தப் பகுதி மேற்கொண்டுவிடும். முதல் வேலையாக இயக்க முறையின் மற்ற எல்லாப் பகுதிகளையும் நினைவகத்தில் எடுத்து வைத்துக் கொள்ளும். இப்போது கணிப்பொறியின் முழுக்கட்டுப்பாடும் இயக்க அமைப்பின்கீழ் வந்து விடும். இயக்க அமைப்பு படிப்படியாக இவ்வாறு கணிப்பொறிக்குள் நுழைவது Booting எனப்படும். முழுக்காலணியில் கயிற்றை சிறிது சிறிதாக இறுக்கிக் கட்டுவது போல் இருப்பதால் இந்தப் பெயர் வந்தது.

இயக்க அமைப்பு கணிப்பொறியின் ஆட்சியை ஏற்றதும், அது நம்முடைய கட்டளைக்காக காத்து நிற்கும். நாம் கட்டளைகள் இட இட அவற்றைச் செய்து முடிக்க உதவும். மின்சாரத்தை நிறுத்தும் வரை தொடர்ந்து இவ்வாறு செய்யும். அதுவரை கணிப்பொறியானது இயக்க அமைப்பின் கட்டுப்பாட்டில்தான் இருக்கும்.

சற்றுப் பெரிய கணிப்பொறிகளில்; ஒரு சமயத்தில் பலருடைய கட்டளை களையும் ஏற்க வேண்டும்; அனைவருக்கும் உதவி செய்ய வேண்டும். ஒவ்வொரு வருடைய மென்பொருள்களையும் தனித்தனியாக வைத்துச் செயல்பட வேண்டும்; அவை ஒன்றுக்கொன்று கலந்து குழப்பம் விளையாமல் காக்க வேண்டும். இவற்றையெல்லாம் செய்யும் இயக்க அமைப்பு சற்று சிக்கலான அமைப்புள்ளதாக இருக்கும். யூனிக்ஸ் (Unix) என்பது இவ்வகையைச் சேர்ந்த ஒரு பிரபல இயக்க அமைப்பு. ஆப்பிள் வகைக் கணிப்பொறிகளை நிருவகிக்க தனி இயக்க அமைப்பு கள் உள்ளன.

நினைவில் நிற்க

1. கணிப்பொறியை நிருவகிக்கும் மென்பொருள் இயக்க அமைப்பு.. இது கணிப்பொறியின் எல்லா பாகங்களையும், அவற்றை இயக்கும் விதிமுறைகளையும் நன்கு அறியும்.

2. கணிப்பொறிக்குள் முதலில் உள்ளே செல்லும் மென்பொருள் இந்த இயக்க அமைப்புதான். படிப்படியாக உள்ளே செல்லும். மின்சாரம் நிறுத்தப் படும் வரை தொடர்ந்து செயல்படும்.

3. நம் மென்பொருள்கள் செயல்படத் தேவையான வசதிகள் எல்லா வற்றையும் செய்து கொடுக்கும். நமக்குத் தேவையான நினைவிடத்தை ஒதுக்குவது, மையச் செயலகத்தைக் கொடுப்பது, உள்ளீடு வெளியீடு செய்வது போன்ற வற்றைச் செய்யும்.

4. IBM இன நுண்கணிப்பொறிகளுக்கு, MS DOS, PC DOS பிரபலமானவை.

5. பல நிரல்களை ஒரே சமயத்தில் நிருவகிக்கும் ஆற்றல் பெற்றது யூனிக்ஸ்.

பயிற்சி வினாக்கள்

1. இயக்க அமைப்பின் தேவை என்ன?
2. இயக்க அமைப்பு என்னென்ன வேலைகளைச் செய்யும்?
3. இயக்க அமைப்பு எவ்வாறு கணிப்பொறிக்குள் செல்கிறது என்பதை விவரி.
4. இயக்க அமைப்பு ஒரு சிறந்த நிருவாகி - எவ்வாறு ?
5. MS DOS, PC DOS மற்றும் Unix இவற்றிற்கு இடையே உள்ளே முக்கிய வேற்றுமை என்ன ?

கணிப்பொறி மொழிகளின் தோற்றமும் தீர்வுமுறைகளும்

3

3-1 பொறிமொழியும் கூட்டுமொழியும்

3-2 உயர்நிலை மொழிகள்

3-3 வழிப்படங்கள்

3-4 அடிப்படைக் கட்டுப்பாட்டு அமைப்புகள்

3-5 போலிக் குறியீடுகள்

3-6 உலாவருதல்

கணிப்பொறி மொழிகள் பொறிமொழி, கூட்டுமொழி, உயர்நிலை மொழி என்று வளர்ச்சி பெற்ற விதத்தை முதலில் காண்போம். உயர்நிலை மொழிகளைப் பயன்படுத்தி கட்டளை நிரல்களை எழுதுவதற்கு முன்னால், கொடுத்த கணக்கை எப்படித் தீர்ப்பது என்பது நமக்குத் தெரிந்திருக்க வேண்டும். தீர்க்கும் முறையை முறைப்படி எழுதத் தெரிந்தால்தான் அதை ஒரு கட்டளை நிரலாக மாற்ற முடியும். தீர்வுமுறைகளை வழிப்படங்களாக அல்லது போலிக் குறியீடுகள் மூலம் எழுதலாம். மிகச்சில அடிப்படை அமைப்புகளைக் கொண்டே பலவிதமான கணக்குகளுக்கும் தீர்வு எழுதுவது எவ்வாறு என்று பார்ப்போம். இவ்வாறு எழுதுவதனால், அதைப் புரிந்து கொள்வது எளிது; அதனால் தவறுகள் பெரும் அளவில் தவிர்க்கப்படும்.

கணிப்பொறி மொழி படிக்கும் பலர் மொழியின் இலக்கணத்தை மட்டும் படித்துவிட்டு, கணக்குகளுக்குத் தீர்வுகாணும் முறை பற்றி அறியாது இருப்பார்கள். அவர்களுக்கு புதிய கணக்குகளுக்குக் கட்டளை நிரல்களை எப்படி எழுதுவது என்பது புரிவதில்லை. இக்குறையைப் போக்க வேண்டியே இப்பகுதி சற்று விளக்கமாக எழுதப்பட்டுள்ளது. தீர்வுமுறைகள் எந்தக் கணிப்பொறி மொழியைச் சார்ந்ததில்லை. அதனால், இவற்றை நன்கு அறிந்து கொள்வது வேறு பல கணிப்பொறி மொழிகளைக் கற்பதற்கும் உறுதுணையாக இருக்கும்.

3.1 பொறிமொழியும் கூட்டுமொழியும்

கணிப்பொறியானது நமக்குத் தேவையானவற்றைக் கணக்கிட்டுத் தர வேண்டுமென்றால், நாம் சொல்வதைப் புரிந்துகொண்டு, அதன்படி செயல்பட்டு, விடையை நமக்குத் தரவேண்டும். இதற்கு, நாம் சொல்வது கணிப்பொறிக்குப் புரியவேண்டும். கணிப்பொறி சொல்வது நமக்குப் புரிய வேண்டும். கணிப்பொறியின் தாய்மொழியில் 0, 1 என்ற இரண்டு குறியீடுகள் மட்டுமே இருக்கின்றன என்று பார்த்தோம். அதன் செயல்பாடுகள் எல்லாமே இந்த இரு குறியீடுகளைக் கொண்டுதான். கணிப்பொறிகள் வடிவமைக்கப்பட்ட ஆரம்ப காலத்தில், அவற்றை வடிவமைத்தவர்கள் அதனுடன் இந்த இரண்டு குறியீடுகளை வைத்தே பேசினர். இந்த இரண்டு குறியீடுகளை வைத்துப் பல மொழிகளையே உருவாக்கினர்! ஒவ்வொரு வித கணிப்பொறிக்கும் தனியாக ஒரு மொழி.

கணிப்பொறியுடன் என்ன பேசப்போகிறோம்?

இந்த எண்ணை இதனுடன் கூட்டு;
இந்த எண்ணை அந்த இடத்தில் வை;
இந்த எண் எதிர்மறை எண்ணா?
இந்தக் குறியீடு பு என்பதா?

- என்பது போல்.

இதுபோன்ற, கணிப்பொறிக்கான கட்டளைகள் (உத்தரவுகள்) சுமார் நூறு இருக்கும். IBM 360 என்ற கணிப்பொறி பேசும் மொழியில் சில எடுத்துக்காட்டுகளைப் பார்ப்போம். இந்தக் கட்டளைகளில் எல்லாமே, என்ன செய்ய வேண்டும் என்பதும், அதற்கான தகவல்கள் எங்கே இருக்கும் என்பதும் இருக்கும். 0101 1010 0100 0000 0000 0000 1000 0000 (பதினாறு நிலை எண்ணாக எழுதினால் 5A400080) என்பதற்கு, நினைவகத்தில் (80_{16} =) 128 ஆம் இடத்தில் உள்ள எண்ணை, நாலாம் பதிவிடத்தில் உள்ள எண்ணுடன் கூட்டி, விடையை மீண்டும் நாலாம் பதிவிடத்தில் வை, என்பது பொருள். இந்தக் கட்டளையின் முதல் பகுதியான 0101 1010 என்பது கூட்டு என்பதைக் குறிக்கும். இதற்கு, எங்கிருந்து தகவல்கள் பெற வேண்டும், செயல் முடிந்தவுடன் விடையை எங்கு வைக்க வேண்டும் என்ற தகவல்கள் அடுத்த இரண்டு பகுதிகளில் உள்ளன.

இதேபோல், 0001 1001 0010 1000 (=1924_{16}) என்றால், இரண்டாம் மற்றும் நான்காம் பதிவிடங்களில் உள்ள எண்களை ஒப்புநோக்கி அவை சமமா அல்லது எது சிறியது என்று பார் - என்று பொருள்.

இதுபோன்று கட்டளைகளை எழுதும்போது, எண்களுக்கிடையில் இடம் விடாது எழுத வேண்டும். நமக்கு எளிதாகப் புரிவதற்காக இங்கு இடம் விட்டு எழுதியிருக்கிறோம். ஆதியில் தோன்றிய ENIAC, EDSAC போன்ற கணிப்பொறி களில் பொறிமொழிகளில்தான் கட்டளைகளை எழுதினார்கள். கணிப்பொறியை வடிவமைக்கும்போதே, இந்தக் கணிப்பொறி இந்த மொழி பேசும் என்று அமைத்து விடுவார்கள். இம்மாதிரி மொழிகள் **பொறிமொழிகள்** (Machine Languages) எனப்படும். இப்பொறிமொழிகளில் உத்தரவுகளை ஒவ்வொன்றாக நாம் எழுத வேண்டுமானால் மிகவும் கடினமானதாக இருக்கும்; தவறுகள் நேர வாய்ப்புகள் நிறைய உண்டு.

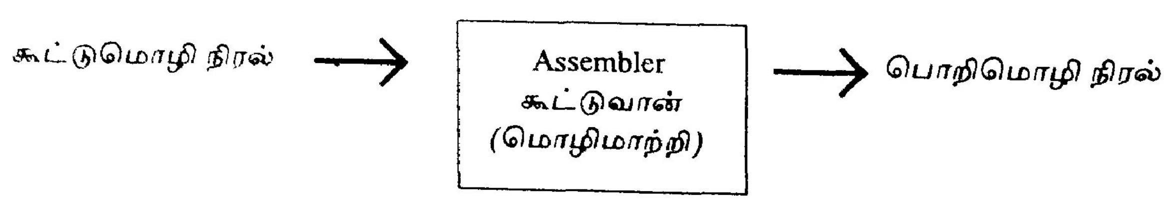
இந்தச் சுமையை ஓரளவு போக்க உருவாக்கப்பட்டதுதான் **கூட்டுமொழி** (Assembly Language). மேலே பார்த்த இரண்டு பொறிமொழி உத்தரவுகளை,

ADD 4, 128 என்றும்

CMP 2, 4 என்றும் எழுதலாம். இப்படி எழுதுவது சற்று சுலபம். எந்தச் செய லுக்கு எந்தக் கோவை என்று நினைவில் நிறுத்த வேண்டிய அவசியமில்லை. மேலும் எண்களை தசம எண்களாக எழுதலாம்.

நமக்குச் சுலபமாக இருந்தாலும் கணிப்பொறிக்கு இந்தப் புதுமொழி புரியாது. கூட்டுமொழியில் எழுதிய கட்டளை நிரல்களை, பொறிமொழியில் மொழி பெயர்த்த பின்னரே, அவற்றைச் செயலாக்க முடியும். கூட்டுமொழியில் உள்ள உத்தரவுகளை பொறிமொழிக்கு மாற்றுவது எளிது.

ADD என்று வந்தால் அதை 0101 1010 என்று மாற்றி எழுது. அதற்குப் பின்வரும் இரண்டு எண்களை இருநிலை எண்களாக மாற்று; -- போன்ற கட்டளைகள் அடங்கிய ஒரு நிரலைக் கொண்டு எளிதில் மொழிபெயர்த்து விடலாம். இந்த மொழிபெயர்ப்பையும் கணிப்பொறியிடமே விட்டுவிடலாம். இந்த மொழிபெயர்க்கும் கட்டளை நிரல் பொறிமொழியில் இருக்கும். அதைச் செயல்படுத்த வேண்டும். அப்போது உள்ளிடும் தகவலாக, நாம் எழுதிய கூட்டுமொழிக் கட்டளை நிரலைத் தரவேண்டும். வெளியீடாக, இந்த நிரல் பொறிமொழியில் வெளிவரும். மொழிமாற்றம் செய்யும் இந்த மென்பொருள் கூட்டுவான் (Assembler) எனப்படும்.



கூட்டுமொழியில் கட்டளை நிரலை எழுதுவதில் மேலும் சில நன்மைகள் உண்டு. ஒரு தகவலை நாம் பெயரிட்டு அழைக்கலாம். அதை நினைவகத்தில் எங்கு வைக்க வேண்டும் என்பதைக் கூட்டுவானே முடிவு செய்து கொள்ளும். அந்த இடத்தையும் நினைவில் வைத்துக் கொள்ளும். இதற்கேற்றபடி பொறிமொழி நிரலைத் தயாரித்துக் கொடுக்கும். மேலே குறிப்பிட்டது போன்ற சில்லறை வேலைகளை நமக்காகக் கூட்டுவான் செய்வதால், நாம் கவனம் சிதறாமல் நம் பணியில் ஈடுபடலாம். ஓர் எடுத்துக்காட்டைப் பார்க்கலாம்.

இன்றைய விற்பனையையும், நேற்றைய விற்பனையையும் கேட்டு வாங்கி, அவற்றைக் கூட்டி மொத்த விற்பனையைக் கணக்கிடு. இதைக் கணிப்பொறி எவ்வாறு செயல்படுத்துமென்று பார்ப்போம். முதலில் இதைச் சற்று விளக்கமாக எழுதுவோம்.

1. இன்றைய விற்பனை எவ்வளவு என்று கேட்டு அந்த மதிப்பை வாங்கு,
2. நேற்றைய விற்பனை எவ்வளவு என்று கேட்டு அந்த எண்ணையும் வாங்கு.
3. மொத்த விற்பனை = இன்றைய விற்பனை + நேற்றைய விற்பனை.

இப்படி எழுதுவது எவ்வளவு எளிது! இன்றைய விற்பனை, நேற்றைய விற்பனை, மொத்த விற்பனை என்று குறிப்பிடப்படும் எண்களை நினைவகத்தில் எங்கே வைத்துக்கொள்வது என்ற விபரங்களை, கூட்டுவானே முடிவுசெய்யும்.

கூட்டுமொழியில் எழுதுவதில் உள்ள சில நன்மைகள்:

1. செயல்பாடுகளைக் குறிக்க பெயர்களைப் பயன்படுத்தலாம்.
   (உதாரணம் - ADD, SUB, MUL).

2. தகவல்களைக் குறிக்கப் பெயர்களைப் பயன்படுத்தலாம் (உதாரணம் மொத்த விற்பனை)

3. தசம எண்களைப் பயன்படுத்தலாம்.

4. தகவல்களை நினைவகத்தில் வைக்கும் இடங்களைக் கூட்டுவானே முடிவு செய்யும்.

ஆனால், இத்துடன் நம் சிக்கல்கள் தீர்ந்துவிடவில்லை. கூட்டுமொழியில் ஒரு சிறு செயலைச் செய்யவும் பல உத்தரவுகளை எழுத வேண்டும். ஓர் எடுத்துக்காட்டைப் பார்ப்போம். இதைப் புரிந்து கொள்ளுங்கள். மனப்பாடம் செய்யாதீர்கள்.

பத்து எண்களை வரிசையாக உள்ளீடு செய்ய வேண்டும் என்போம், இதற்காகச் செய்ய வேண்டிய செயல்கள் பல. 10, 9, 8,.... என்று எண்ணிக் கொண்டே வரவேண்டும். 0 என்று வந்தவுடன் நிறுத்த வேண்டும். ஒவ்வொரு முறையும் இதற்காக 'எண்ணிக்கையில்' 1 என்ற எண்ணைக் கழிக்க வேண்டும்.

அதனால் 'எண்ணிக்கை' என்பதை ஓர் இடத்திலும், 1 என்பதை இன்னொரு இடத்திலும் வைத்திருக்க வேண்டும். அடுத்தது, உள்ளீடாக வரும் எண்ணை எங்கே வைக்க வேண்டும் என்று கூற ஒரு முகவரியை நினைவில் வைக்க வேண்டும். ஒவ்வொரு முறையும், இந்த முகவரியை மாற்ற வேண்டும். வரும் எண் ஒரு பின்னம் என வைத்துக்கொண்டால், அதற்கு 4 பைட்டுகள் தேவை. அதனால் ஒவ்வொரு முறையும் இருக்கும் முகவரியுடன் 4 சேர்த்து புது முகவரியைக் கணக்கிட வேண்டும். இதற்கு 'முகவரி' என்பதையும், 4 என்பதையும் நினைவில் வைக்க வேண்டும். 'எண்ணிக்கை', 'முகவரி', 1, 4 என்பவற்றை அடிக்கடி பயன்படுத்து வதால் அவற்றை பதிவிடங்களிலேயே வைப்பது நல்லது. நமக்குத் தேவையான செயல்பாட்டை இப்படி எழுதலாம்.

பதிவிடம் இரண்டில் 10 என்று எழுது. (ஆரம்ப எண்ணிக்கை = 10)

பதிவிடம் மூன்றில் 1 என்று எழுது.

பதிவிடம் நான்கில் 8000 என்று எழுது. (ஆரம்ப முகவரி = 8000)

பதிவிடம் ஐந்தில் 4 என்று எழுது.

படிப்பது : ஒரு எண்ணைப் படித்து, அதை நான்காம் பதிவிடம் குறிப்பிடும் முகவரியில் வை.

பதிவிடம் இரண்டிலிருந்து பதிவிடம் மூன்றில் உள்ள எண்ணைக் கழி.

பதிவிடம் நான்குடன் பதிவிடம் ஐந்தில் உள்ள எண்ணைக் கூட்டு.

பதிவிடம் இரண்டில் 0 என்றால் வேலை முடிந்துவிட்டது; இல்லையென்றால் **படிப்பது** என்னும் இடத்திற்குப் போய் திரும்பவும் செய்.

ஒரு சிறு செயலைச் செய்ய பதிவிடங்களில் என்ன இருக்கின்றன என்பதையும் நாமே நினைவில் வைத்துக்கொள்ள் வேண்டும். இதைக் கவனித்தீர்களா? படிப்பது என்பதை ஒரு பத்தியின் தலைப்பாகப் பயன்படுத்தி உள்ளோம். இந்தத் தலைப்பும் ஒரு முகவரிபோல் அடையாளம் காட்டப் பயன்படுகிறது.

இன்று எழுதப்படும் கட்டளை நிரல்கள் மிகப்பெரியவை. கூட்டு மொழியில் எழுதினால் பல லட்சக்கணக்கான வரிகள் வரும். அப்படி எழுதுவது மிகவும் கடினம். அதனால் அப்படி எழுதுவதில்லை. உயர்நிலை மொழிகளில்தான் எழுதுகிறார்கள். அதுபற்றி அடுத்த பாடத்தில் பார்ப்போம்.

நினைவில் நிற்க

1. ஒவ்வொரு கணிப்பொறி உருவாகும்போதும், அதற்கு ஒரு பொறிமொழி தாய்மொழியாக உருவாக்கப்படுகிறது. அதில் 0, 1 என்ற இரண்டு குறியீடுகள் மட்டுமே உண்டு. அதில் மனிதர்கள் எழுதுவது கடினம்.

2. கூட்டுமொழி பொறிமொழியை விடச் சுலபமானது. கூட்டுவான் என்னும் மென்பொருள், கூட்டுமொழியில் எழுதப்பட்ட கட்டளை நிரலை பொறிமொழிக்கு மாற்றித்தரும். கூட்டுவான் நமக்காகப் பல சில்லறை வேலைகளைச் செய்வதால், கூட்டுமொழியில் எழுதுவது சற்று எளிது.

3. பெரிய கட்டளை நிரல்களைக் கூட்டுமொழியில் எழுதுவது கடினம்.

பயிற்சி வினாக்கள்

1. பொறிமொழிக்கும் கூட்டு மொழிக்கும் உள்ள வேற்றுமைகள் நான்கினைக் கூறு.
2. கூட்டுவான் என்றால் என்ன? அதன் பணி என்ன?
3. கூட்டுவான் நமக்காகச் செய்யும் வேலைகள் நான்கினைக் கூறு.
4. கூட்டுமொழியில் கட்டளை நிரலை எழுதுவதன் நன்மைகள் நான்கினைக் கூறு.
5. ஒரு சிறு செயலைச் செய்ய வேண்டுமானாலும் அதற்கு பல கட்டளைகள் தேவைப்படலாம் என்பதை ஓர் உதாரணத்துடன் விளக்கு.

3.2 உயர்நிலை மொழிகள்

'பத்து எண்களைப் படித்து அவற்றை நினைவில் வை' - இப்படிக் கூறுவது தான் எவ்வளவு எளிதாக இருக்கிறது? எண்ணிக்கையையும் முகவரிகளையும் நினைவில் வைத்துக் கொள்வது, அவற்றைத் தேவையானபடி மாற்றுவது, எண்ணிக்கை தேவையான அளவு வந்து விட்டதா என்று பார்ப்பது போன்ற வேலைகளையும் நமக்காக யாராவது செய்தால், நமக்கு எளிதாக இருக்கும். இப்படிச் செய்வதற்கான முயற்சியில் முதல் கட்டம்தான் ஃபோர்ட்ரான் (FOR TRAN - Formula Translation) என்னும் **உயர்நிலை மொழியும்** (High Level Langage), இந்த மொழியில் எழுதப்பட்ட கட்டளை நிரல்களை பொறி மொழிக்கு மாற்றம் செய்யும் **தொகுப்பான்** (Complier) என்ற ஒரு மொழிமாற்றியும்.

ஃபோர்ட்ரான்

இம்மொழி முதலில் 1954 இல் வடிவமைக்கப்பட்டு, 1957 இல் செயல்படுத்தப்பட்டது. பின்னர் சிற்சில மாற்றங்களுடன் பல வெளியீடுகள் வந்துள்ளன. இவற்றில் புகழ்பெற்றவை FORTRAN II, FORTRAN IV மற்றும் FORTRAN 77. இம்மொழியை வடிவமைப்பதில் முக்கிய பங்கேற்றவர் பேகஸ் (Backus) என்பவர்.

வழிமுறைகளைச் சிறந்த முறையில் எழுதுவதற்காகவே வடிவமைக்கப்பட்ட மொழி **ஆல்கால்** (ALGOL - Algorithmic Language). இது ஒரு குழுவால் வடிவமைக்கப்பட்டு, 1958, 1960 மற்றும் 1968 ஆம் ஆண்டுகளில் பல பதிப்புகளாக வெளிவந்தது. ஃபோர்ட்ரானும், ஆல்காலும் அறிவியல் கணிப்பிற்கே பெரிதும் பயன்பட்டன.

அறிவியல் துறைக் கணிப்பில் கூட்டல், பெருக்கல் போன்ற கணிதச் செயல்பாடுகள் மிகுதியாகவும், உள்ளீடும் வெளியீடும் மிகக் குறைவாகவும் இருக்கும். ஆனால் வணிகத்துறைக் கணிப்புகளில் இதற்கு நேர்மாறாக இருக்கும். உள்ளீடும் வெளியீடும் அதிகமாகவும், கணிதச் செயல்பாடுகள் குறைவாகவும் இருக்கும். எடுத்துக்காட்டாக, ஒரு விற்பனைச் சீட்டு தயாரிப்பதற்குச் செய்ய வேண்டிய கணக்குகள் குறைவு; வெளியீடு செய்வதற்குத்தான் அதிக நேரமாகும். வணிகத்துறைக்கென்றே உருவாக்கப்பட்ட மொழி **கோபால்** (COBOL - Common Business Oriented Language). இதன் முதல் வடிவம் 1959 இல் தோன்றியது. இதுவும் ஒரு குழுவால் வடிவமைக்கப்பட்டது.

பேசிக்

கற்றுக் கொள்ளவும், பயன்படுத்தவும் எளிமையாக இருக்க வேண்டும் என்னும் நோக்கத்துடன் உருவாக்கப்பட்டது **பேசிக்** (BASIC - Beginners All Purpose Symbolic Instruction Code). 1965 இல் டார்ட்மௌத் கல்லூரியில், ஜான் கெம்னி, டாம் கர்ட்ஸ் (John Kemney and Tom Kurtz) என்பவர்களால் வெளியிடப்பட்டது. கணிப்பொறியின் முன் அமர்ந்து, அதனுடன் நேரடியாக உரையாடும் வகையில் அமைந்த முதல் உயர்நிலை மொழி இதுதான். நுண் கணிப்பொறிகள் வர ஆரம்பித்தபோது, அதனுடன் சேர்ந்து பேசிக்கும் அதிக அளவில், படிக்க மட்டும் நினைவகத்தில் பொறிக்கப்பட்டு வெளிவந்தது. இதற்குக் காரணம் பேசிக்கின் சிறிய அளவும், எளிமையுமே.

அறிவியல், வணிகத்துறை என்று இரண்டுக்குமே பயன்படும் வகையில் ஒரு பொது மொழி IBM ஆதரவில் 1965-66 இல் உருவானது. இது PL-I (Programming Language-One). இதன் பெரிய அளவே இதன் மாபெரும் குறையாக அமைந்தது. அதனால், அதிகம் பிரபலமாகவில்லை.

எதிலும் முன்னோடியாக இருப்பதில் நன்மையும் உண்டு, தீமையும் உண்டு. எளிதில் பிரபலம் அடையலாம். ஆனால், பின்னால் வருபவர்கள் "இதை இப்படிச் செய்திருக்கலாம், அது இப்படி இருக்கக் கூடாது" என்று குறைகள் பல கூறுவார்கள். ஃபோர்ட்ரானும், பேசிக்கும் இதற்கு விதி விலக்கல்ல. இன்றும் அவை பிரபலமாக இருந்தாலும், அவற்றின் குறைகள் பல தெரியத் தொடங்கின. பொறிமொழி, கூட்டுமொழிபோல் அவ்வளவு அதிகமாக இல்லாவிட்டாலும், நம் கவனத்தைச் சிதறவைக்கும் பல விஷயங்கள் இவற்றில் இருப்பதாகவும், சிக்கலான செயல்களைச் செய்யும் ஆற்றல் இவற்றுக்கு இல்லை என்றும் கூறப்பட்டது. பெரிய கட்டளை நிரல்களை வடிவமைப்பதற்கு இவை தடையாக இருக்கும் எனக் கருதப்பட்டது.

பாஸ்கல்

கட்டளை நிரல்களின் வடிவத்தில், ஓர் ஒழுங்கு, ஒரு கட்டமைப்பு தேவை. இவ்வொழுங்கைப் பின்பற்றி எழுதும் நிரல்களை வடிவமைப்பதும் எளிது; புரிந்துகொள்வதும் எளிது-என்று அனுபவ உண்மைகளை அடிப்படையாக வைத்து பாஸ்கல் என்னும் மொழி உருவாக்கப்பட்டது (Pascal ஒரு பிரெஞ்சுக் கணித மேதை). இது 1968 இல் முதலில் வெளிவந்தது. இதன் முதல் மொழிமாற்றி 1970 இல் வந்தது. ஜென்ஸன், விர்த் (Kathleen Jansen and Nicklaus Wirth) என்பவர்களால் உருவக்கப்பட்டது இந்த மொழி. கணிப்பொறி இயலைக் கற்பிப்பதற்காகவே ஏற்படுத்தப்பட்ட இந்த மொழி ஆல்காலின் ஒரு வழித் தோன்றல்.

ஸி

மனிதன் உயர்நிலை மொழிகளைப் புரிந்துகொண்டு அவற்றில் எழுதுவது எளிது. கணிப்பொறி பொறிமொழியில்தான் வேலை செய்யும். அதனால், பொறிமொழியானது பொறிக்கு அருகிலும், உயர்நிலை மொழி மனிதனுக்கு அருகிலும் இருப்பதாக நினைக்கலாம். கூட்டுமொழியானது இரண்டுக்கும் நடுவில், மனிதனைவிடப் பொறிக்குச் சற்று அருகில் இருப்பதாக நினைக்கலாம். உயர்நிலை மொழியாகவும், அதே சமயத்தில் பொறிக்குச் சற்றுப் பக்கத்திலும் உள்ள ஒரு மொழி ஸி (C). இதனை வடிவமைத்தவர் டென்னிஸ் ரிச்சி (Dennis Ritchie). இதில் வாக்கியங்கள் மிகவும் சுருக்கமாக இருக்கும்; ஆனால் மிகவும் சக்தியுள்ளவையாக இருக்கும். பல்வேறு சிக்கலான கணிப்புகளைச் செய்ய இம்மொழி பெரிதும் பயன்படுகிறது. இயக்க முறைகள், மொழிமாற்றிகள் போன்றவை இந்த மொழியில் எழுதப்படுகின்றன.

இங்கு ஒரு சில மொழிகளைப் பற்றியே பார்த்தோம். ஆனால், இன்று சில நூறு மொழிகள் உள்ளன! கணிப்பொறி மொழிகள் பற்றிய அறிவு இப்போது வளர்ச்சி பெற்றுள்ளது. புதுப்புது உத்திகள் கண்டுபிடிக்கப்பட்டு, அவற்றைப் பயன் படுத்த புதுப்புது மொழிகள் உருவாக்கப்படுகின்றன. இவ்விதத்தில் C இன் வழித்தோன்றல் C + +. விரைந்து பிரபலமாகி வரும் மொழி இது. **ஏடா** (Ada) என்னும் மொழி அமெரிக்க அரசின் ஆதரவு பெற்றுப் பாதுகாப்புத் துறைக்காக உருவாக்கப்பட்டது. **லிஸ்ப்** (LISP - List Processing), **ப்ரொலாக்** (Polog- - Programming in Logic) போன்ற மொழிகள், செயற்கை அறிவைச் செயல்படுத்து வதற்கான ஐந்தாம் தலைமுறை மொழிகள்.

வணிகத்துறையில், **உறவுநிலைத் தகவல் அமைப்பைப்** (Relational Data Base) பயன்படுத்தும் **ஆரகிள்** (Oracle), **இங்ரஸ்** (Ingres), **யூனிஃபை** (Unify), **சைபேஸ்** (Sybase) போன்றவை நான்காம் தலைமுறை மொழிகள்.

நாம் ஏற்கனவே பார்த்த உயர்நிலை மொழிகள் மூன்றாம் தலைமுறை, கூட்டு மொழிகள் இரண்டாம் தலைமுறை. பொறிமொழிகள் முதல் தலைமுறை மொழிகள்.

மொழிமாற்றிகள்

ஒவ்வொரு புது உயர்நிலை மொழி உருவாக்கப்படும்போதும், அதில் எழுதப்படும் கட்டளை நிரல்களை பொறிமொழிக்கு மாற்றம் செய்ய ஒரு மொழிமாற்றியும் தேவை. இந்த மொழிமாற்றிகளுக்குக் கூட்டுவான்களை விட வேலை மிக அதிகம். ஏனென்றால் உயர்நிலை மொழிகளில் ஒன்றிரண்டு வரிகளில் சொல்வதை, பொறிமொழியில் பல வரிகளாக மாற்றம் செய்ய வேண்டும். பலவிதச் சில்லறை வேலைகளையும் செய்து கொடுக்க வேண்டும்.

மொழிமாற்றிகளில் இரண்டு வகை உண்டு. முதல் வகை, உள்ளீடாக வரும் உயர்நிலை மொழிக் கட்டளை நிரலை முழுவதும் வாங்கிக் கொண்டு, மொத்தத்தையும் ஒரே சமயத்தில் மொழி மாற்றம் செய்து, பொறிமொழி நிரலை முழுவதுமாக வெளியீடு செய்யும். இந்த வகை மொழிமாற்றி **தொகுப்பான்** (Compiler) எனப்படும். உயர்நிலை மொழியில் உள்ள கட்டளை நிரல் **மூலநிரல்** (Source code) எனப்படும். மொழிமாற்றம் செய்யப்பட்டு பொறிமொழியில் கிடைக்கும் நிரல் **இலக்குநிரல்** (Object code) எனப்படும். மொழிமாற்றம் செய்தவுடன் தொகுப்பானின் வேலை முடிந்துவிடும். மூல நிரலின் தேவையும் முடிந்துவிடும். இனி, கணிப்பொறியில் செயல்படுத்தத் தேவையானது இலக்கு நிரல் மட்டும்தான். இந்த முதல் வகை மொழிமாற்றம், தமிழ் தெரியாத ஒருவர் மேடையில் தன் மொழியில் முழுவதுமாகப் பேசி முடித்ததும் மொழிபெயர்ப்பாளர் மொத்தமாக மொழிபெயர்த்துத் தமிழில் கூறுவதற்கு ஒப்பாகும்.

அடுத்தவகை, பேச்சாளர் ஒவ்வொரு வக்கியத்தை முடித்ததும், உடனுக்குடன் அந்த வாக்கியத்தை மொழிபெயர்ப்பது. இந்த வகையில், கட்டளை நிரலின் வரிகள் ஒவ்வொன்றாக எடுக்கப்பட்டு பொறிமொழியில் மாற்றம் செய்யப்படும். இந்தப் பொறிமொழி உத்தரவுகள் உடனடியாகச் செயல்படுத்தப்படும். செயல்படுத்தி முடிந்தவுடன் அடுத்த வரி எடுத்து மொழி பெயர்க்கப்படும். இந்த வகை மொழிமாற்றி **வரி-மொழிமாற்றி** (Interpreter) எனப்படும். பேசிக் மொழிக்கு பெரும்பாலும் வரி-மொழிமாற்றிதான் உண்டு. பெரும்பாலான உயர்நிலை மொழிகளுக்கு தொகுப்பான்களே கிடைக்கும்.

வரி-மொழிமாற்றியில் நான்கு குறைகள் உண்டு. ஒன்று கணிப்பு முடியும் வரை மூல நிரலும், வரி-மொழிமாற்றியும் கணிப்பொறியின் நினைவில் இருக்க வேண்டும். இரண்டு - ஒரு வரியைத் திரம்பத்திரும்ப பல முறை செய்ய வேண்டி வந்தால் ஒவ்வொரு முறையும் அந்த வரியை மொழிமாற்றம் செய்ய வேண்டும். மூன்று - இலக்க நிரல் உருவக்கப்படுவதில்லை. அதனால் நிரல் செயல்படும் ஒவ்வொரு முறையும் மொழிமாற்றம் செய்யப்பட வேண்டும். நான்கு - நிரல் செயல்படும்போது மொழி மாற்றமும் செய்ய வேண்டியிருப்பதால் செயல்படும் நேரம் அதிகமாகும்.

நினைவில் நிற்க

1. ஒவ்வொரு வகைக் கணிப்பொறிக்கும் தாய்மொழியாக ஒரு பொறிமொழி உண்டு. கணிப்பொறி இந்த மொழியைக் கொண்டுதான் வேலை செய்யும்.

2. பொறிமொழியில் எழுதுவதை விடக் கூட்டுமொழியில் எழுதுவது எளிது.

3. ஃபோர்ட்ரான், பேசிக், ஆல்கால், கோபால் போன்றவை முதலில் தோன்றிய உயர்நிலை மொழிகள்.

4. ஆல்காலின் வழித்தோன்றலான பாஸ்கல் கட்டமைப்புக்கு முக்கியத்துவம் கொடுத்த மொழி.

5. உயர்நிலை மொழி என்றாலும், கணிப்பொறிக்கு அருகிலும் இருப்பது ஸி. சக்தி வாய்ந்த மொழி. சிக்கலான மென்பொருட்களை உருவாக்க உதவுகிறது.

6. வணிகத்துறையில் பயன்படும் நான்காம் தலைமுறை மொழிகள் ஆரகிள், இங்ரஸ், யூனிஃபை, சைபேஸ் போன்றவை.

7. செயற்கை அறிவைச் செயல்படுத்த உதவும் ஐந்தாம் தலைமுறை மொழிகள் லிஸ்ப், ப்ரொலாக் போன்றவை.

8. உயர்நிலை மொழி மாற்றிகளில் தொகுப்பான், வரி-மொழி மாற்றி என இரண்டு வகை உண்டு. கூட்டு மொழியிலிருந்து மாற்றுவது கூட்டுவான்.

பயிற்சி வினாக்கள்

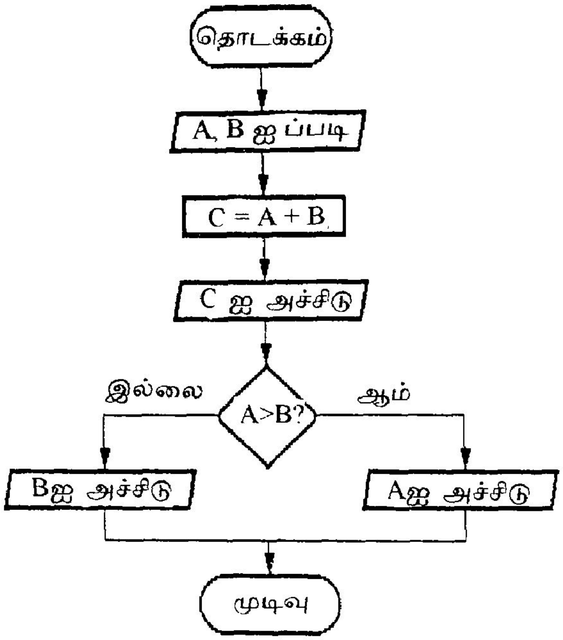
1. மொழி மாற்றியின் தேவை என்ன?
2. முதலில் தோன்றிய உயர்நிலை மொழி எது? அதன் நோக்கம் என்ன?
3. முதலில் தோன்றிய உரையாடல் வகை மொழி எது?
4. பேசிக் மொழியின் சிறப்புகள் இரண்டினைக் கூறு.
5. பாஸ்கல் மொழி எதற்காக உருவாக்கப்பட்டது?
6. ஸி மொழியின் சிறப்பு என்ன?
7. வணிகத்துறையில் இன்று பயன்படும் மொழிகள் சிலவற்றைக் கூறு.
8. செயற்கை அறிவுத்துறையில் பயன்படும் மொழிகள் இரண்டினைக் கூறு.
9. மொழிமாற்றியின் இரண்டு வகைகள் எவை? அவற்றுக்குள்ள வித்தியாசங்கள் என்ன?
10. மூலநிரலுக்கும் இலக்கு நிரலுக்கும் உள்ள வித்தியாசங்கள் இரண்டினைக் கூறு.
11. வரி - மொழிமாற்றியின் குறைபாடுகள் நான்கினைக் கூறு.

3.3 வழிப்படங்கள்

நாம் செய்யும் கணிப்புகள், பயன்படுத்தும் மொழியைப் பொறுத்தது இல்லை; பயன்படுத்தும் கணிப்பொறியைப் பொறுத்ததும் இல்லை. எந்த மொழியைப் பயன்படுத்துவது என்பது, நமக்கு எந்த மொழி தெரியும், நம்மிடம் எந்த மொழிமாற்றி உள்ளது, இந்தச் செயலுக்கு இந்த மொழி பொருத்தமானதா போன்ற விஷயங்களைப் பொறுத்தது. செய்யவேண்டிய செயல் பற்றி, முதலில் தமிழ், ஆங்கிலம் போன்ற மொழிகளில்தான் சிந்திக்கிறோம். பிறகுதான் அதை ஒரு கட்டளை நிரலாக, ஓர் உயர்நிலைக் கணிப்பொறி மொழியில் எழுதுகிறோம்.

கணிப்பொறி மொழிகளில் எழுதப்படும் ஒவ்வொரு வாக்கியமும் தெளிவாக, துல்லியமாக, எந்தவிதக் குழப்பத்திற்கும் இடம் கொடுக்காமல் இருக்க வேண்டும். அதனால் கட்டளை நிரலை உருவாக்கும்போது ஒவ்வொரு வாக்கியத்தையும் நன்கு சிந்தித்து எழுத வேண்டும். தமிழ், ஆங்கிலம் போன்ற இயற்கை மொழிகளில் நாம் எழுதும் வாக்கியங்கள் மிகவும் நீளமாக இருக்கும்; துல்லியமாக இருக்கும் என்று கூற முடியாது. ஏனென்றால் இயற்கையாகத் தோன்றிய எல்லாமொழிகளிலும் அங்கங்கே குழப்பங்கள் இருக்கும். அதனால் வழிமுறைகளை இயற்கை மொழி களில் எழுதுவதைவிட, குழப்பமில்லாத, எளிதான, சுருக்கமான ஓர் இடைநிலை மொழியில் அல்லது குறியீடுகளில் எழுதுவது நல்லது. இதைப்புரிந்து கொள்வதும், இதைப்பார்த்து ஓர் உயர்நிலை மொழியில் கட்டளை நிரலை எழுதுவதும் சுலபமாக இருக்கும். இவ்வாறு இயற்கை மொழிக்கும், உயர்நிலைக் கணிப்பொறி மொழிக்கும் இடையில் இருக்கும் குறியீடுகளில் பிரபலமானவை இரண்டு. அவை **வழிப்படம்** (Flow chart) மற்றும் **போலிக்குறியீடு** (Pseudocode).

வழிப்படங்கள் பற்றி இந்தப் பாடத்தில் பார்ப்போம். ஒரு செயலைச் செய்துமுடிக்க எப்படிச் செயலாற்ற வேண்டும் என்பதனைப் படங்கள் மூலம் குறிப்பிடுவதால் இந்தப் பெயர் வந்தது. முதலில் ஓர் உதாரணத்தைப் பார்ப்போம். இரண்டு எண்களைப் படித்து, அதில் பெரிய எண்ணையும், அவற்றின் கூட்டுத் தொகையையும் அச்சிட வேண்டும். இதைச் செய்ய முதலில் இரண்டு எண்களைப் படித்து அவற்றை இரண்டு பெயர்களின் மதிப்பாக வைக்க வேண்டும். அடுத்தது, அவற்றின் கூட்டுத்தொகையைக் கணக்கிட்டு, அதை அச்சிட வேண்டும். அடுத்து, இரண்டு எண்களில் எது பெரியது என்பதைத் தீர்மானிக்க வேண்டும். பிறகு பெரிய எண்ணை அச்சிட வேண்டும். அத்துடன் வேலை முடியும். இதைக்குறிக்கும் வழிப்படம் படம் 3.1 இல் உள்ளது.



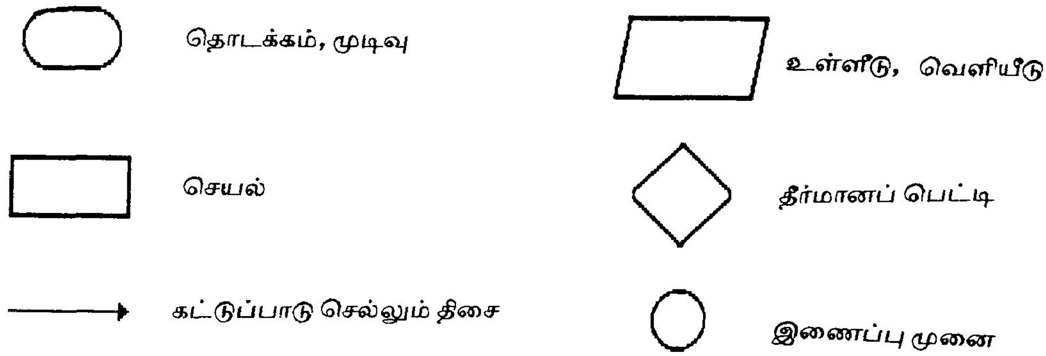
படம் 3 - 1 வழிப்படம்

வழிப்படங்களில் படக்குறியீடுகளை, அவற்றுக்கே உரிய அர்த்தத்துடன் பயன்படுத்துகிறோம். அவை படம் 3.2 இல் கொடுக்கப்பட்டிருக்கின்றன.

முதல் வழிப்படத்தில் C = A + B என்று எழுதியதற்கு ஒரு தனி அர்த்தம் உண்டு. சமன்பாட்டுக்குறியை இங்கு பயன்படுத்தினாலும் அது சமன்பாட்டைக் குறிப்பதற்கு அல்ல. இந்தக் குறியீட்டின் வலது பக்கத்தில் உள்ளதின் மதிப்பைக் கணக்கிட்டு, அதை இடதுபுறத்தில் உள்ள குறியீட்டின் மதிப்பாக வை என்ற பொருளில் பயன்படுத்துகிறோம். இந்ப் புதிய விளக்கத்தின்படி, A = A + 1 என்று எழுதினால் தவறே இல்லை. ஏனென்றால் இது சமன்பாடில்லை. A இன் மதிப்பை எடுத்து, அதனுடன் 1 கூட்டி, மீண்டும் அதே இடத்தில் வை என்று பொருள். அதாவது A இன் மதிப்பு 1 கூடுகிறது. A = A + B என்று எழுதலாம். ஆனால் A + B = A என்று எழுதக்கூடாது. ஏனென்றால் இடதுபுறம் ஒரு பெயர்மட்டும்தான் வரலாம்.

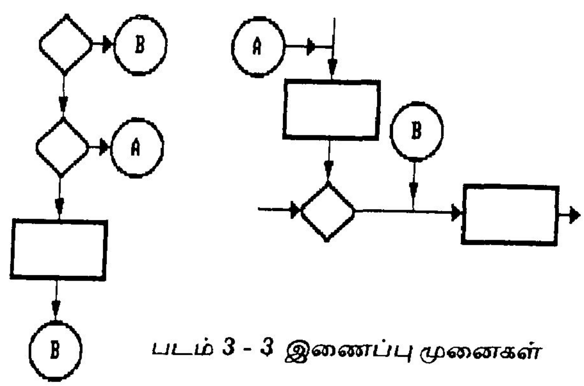
நம் கணிப்பிற்கான வழிமுறைகள் அத்தனையையும் இந்தக் குறியீடுகளைக் கொண்டே எழுதிவிட முடியும். வழிமுறை சின்னதாக இருந்தால், வழிப்படம் ஒரு பக்கத்திற்குள் அடங்கும். வழிமுறை பெரியதாக இருந்தால்? வழிப் படத்திற்குப்

பல பக்கங்கள் தேவை. ஒரு பக்கத்திலிருந்து வேறொரு பக்கத்திற்கு எவ்வாறு அம்புக் குறியால் இணைப்பது? இதைச் சமாளிக்கத்தான் சிறு வட்டமான 'இணைப்பு' என்னும் குறியைப் பயன்படுத்துகிறோம்.



படம் 3 - 2 வழிப்படக் குறியீடுகள்

ஒரே பெயர் கொண்ட எல்லா இணைப்பு வட்டங்களும் ஒரே இடத்தைக் குறிக்கும்; இவை ஒரே பக்கத்திலோ, அல்லது, பல பக்கங்களிலோ இருக்கலாம். படம் 3.3. இல் ஓர் உதாரணத்தைக் காணலாம்.



படம் 3 - 3 இணைப்பு முனைகள்

வழிப்படங்களின் நன்மைகள்

1. துல்லியமானவை. சிறிய படங்கள் நம் எண்ணங்களைத் தெளிவாக வெளிப்படுத்தும்.

2. சிறிய படங்களைப் பார்த்துப் புரிந்துகொள்வது எளிது.

குறைபாடு

பெரிய வழிப்படங்கள் பல பக்கங்களில் இருக்கும். இதைப் புரிந்துகொள்வது கடினம்.

நாம் பயிற்சிக்காகக் கல்வி நிலையங்களில் எடுத்துக் கொள்ளும் கணக்குகள் சிறியவை. அவற்றின் வழிப்படங்களும் சிறியவை. சில உதாரணங்கள் இங்கே கொடுக்கப்பட்டுள்ளன. அவற்றைப் புரிந்துகொண்டு, அவை தேவையான கணிப்பைச் செய்யுமா என்று பாருங்கள்; இந்தப் படங்களை மனப்பாடம் செய்யாதீர்கள். ஒவ்வொரு கணிப்பிற்கும் என்னென்ன செய்ய வேண்டும் என்று எவ்வளவு சிந்திக்க வேண்டியிருக்கிறது என்று கவனியுங்கள்.

1. ஒரு பெட்டியின் நீள, அகல, உயரத்தைப் படித்து அதன் கொள்ளளவைக் கண்டுபிடி.

2. சுழி முதல் நான்கு வரை இருக்கக் கூடிய ஓர் எண்ணைப் படித்து அதை எழுத்தால் எழுது.

3. மூன்று எண்களைப் படித்து அவற்றில் சிறியதை அச்சிடு.

4. $ax^2 + bx + c = 0$ என்ற சமன்பாட்டிற்குத் தீர்வு காண். இதற்கு a, b, c என்ற எண்களைக் கேட்டுப்பெறு.

5. நூறு எண்களைப் படித்து அவற்றில் சிறியதை அச்சிடு.

6. முதல், கூட்டு வட்டி விகிதம் - இவற்றைப் படித்து, ஒவ்வொரு வருட முடிவிலும் கிடைக்கும் தொகை எவ்வளவு என்று பட்டியலிடு.

7. கொடுக்கப்படும் ஓர் எண் பகுபடா எண்ணா என்று கண்டுபிடி.

8. ஒன்று முதல் ஆயிரம் வரை உள்ள பகுபடா எண்களைக் கண்டுபிடித்து எழுது.

9. 1, 2, 3......... என்று எத்தனை எண்களைக் கூட்டினால், ஆயிரம் அல்லது அதற்குச் சற்றுக் கூடுதலாக வரும்?

10. கொடுக்கப்பட்ட முழு எண்ணில் உள்ள இலக்கங்களைக் கண்டுபிடி.

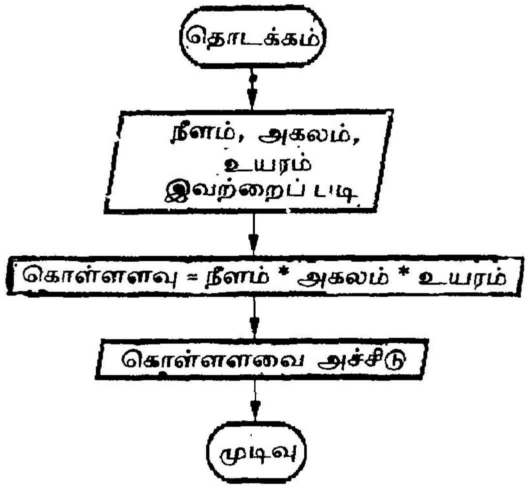
இதற்கான வழிப்படங்கள் அடுத்த சில பக்கங்களில் உள்ளன.

நினைவில் நிற்க

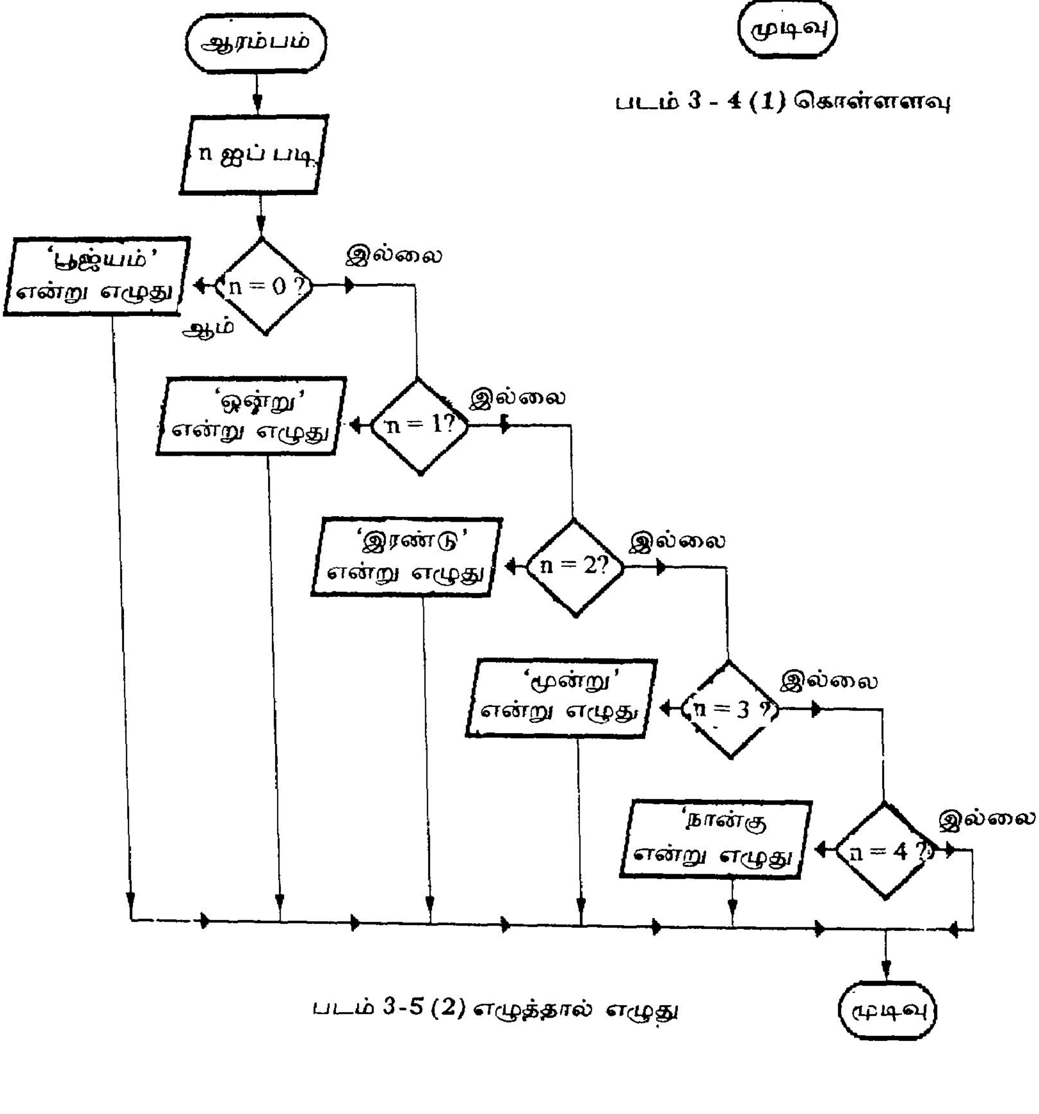
1. சிறிய வழிமுறைகளை வழிப்படங்கள் மூலம் தெளிவாகக் கூறலாம்.

பயிற்சி வினாக்கள்

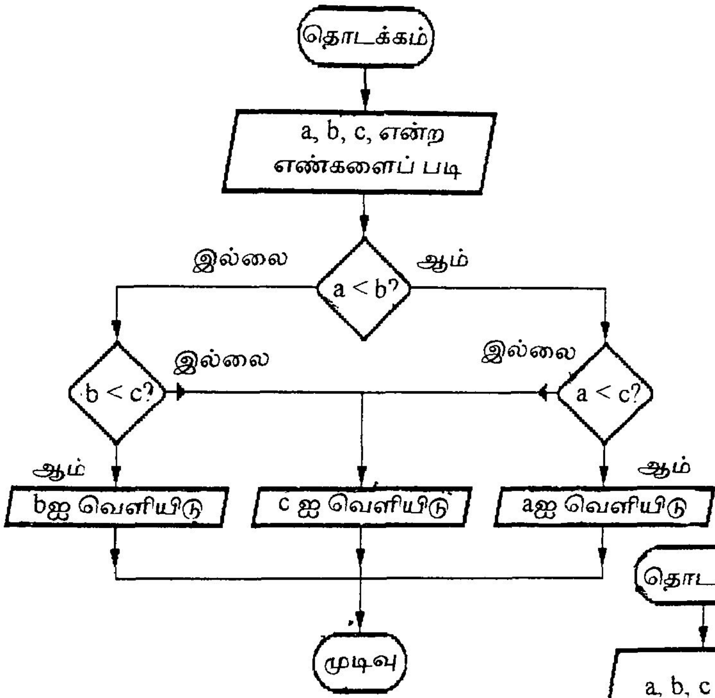
1. வழிப்படத்தில் பயன்படும் குறியீடுகளையும் அவற்றின் பொருளையும் பட்டியலிடு.
2. இணைப்பு முனை என்றால் என்ன? விவரி.
3. வழிப்படங்களின் குறை நிறைகளைக் கூறு.
4. இன்றைய மாலைப்பொழுதை எப்படிக் கழிப்பது என்ற தலைப்பில் ஒரு வழிப்படம் வரை.



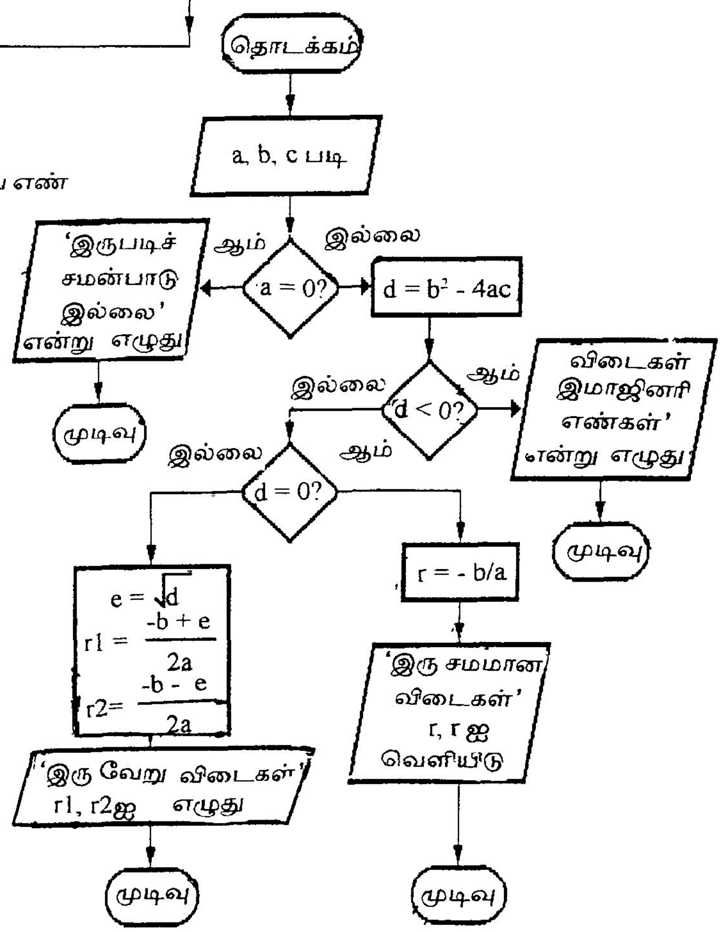
படம் 3 - 4 (1) கொள்ளளவு



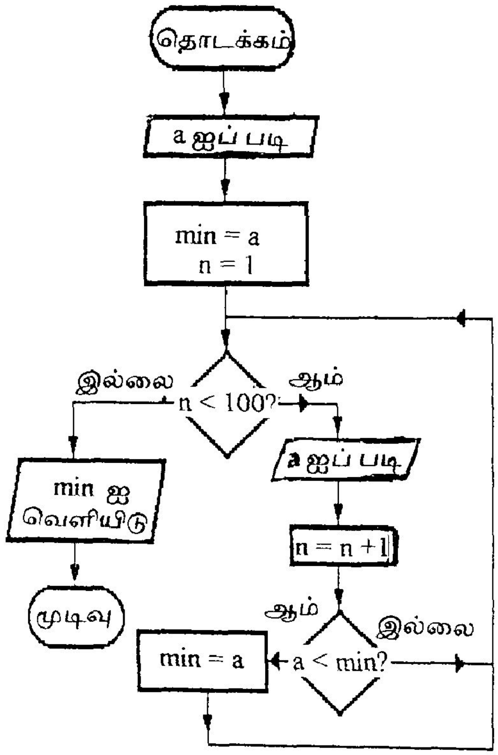
படம் 3-5 (2) எழுத்தால் எழுது



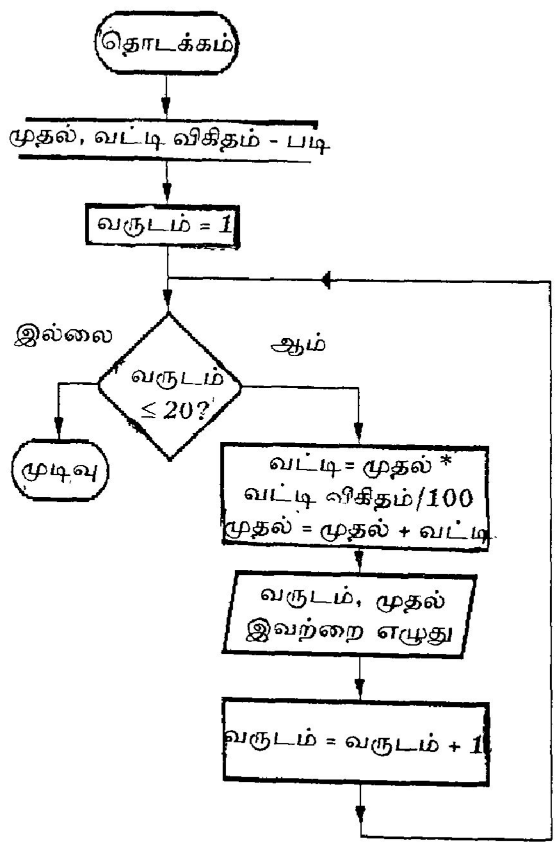
படம் 3 - 6 (3) சிறிய எண்



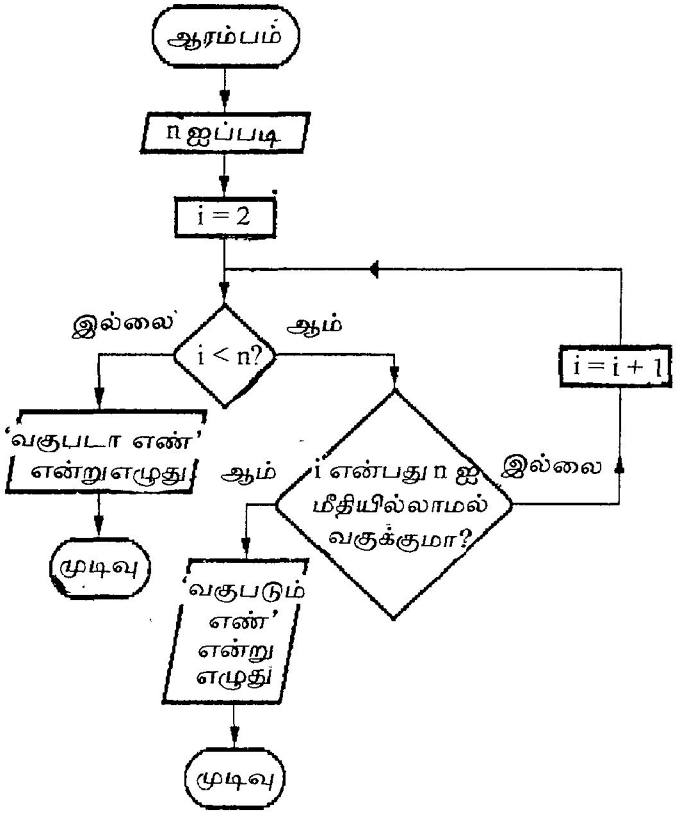
படம் 3 - 7 (4) இருபடிச் சமன்பாடு



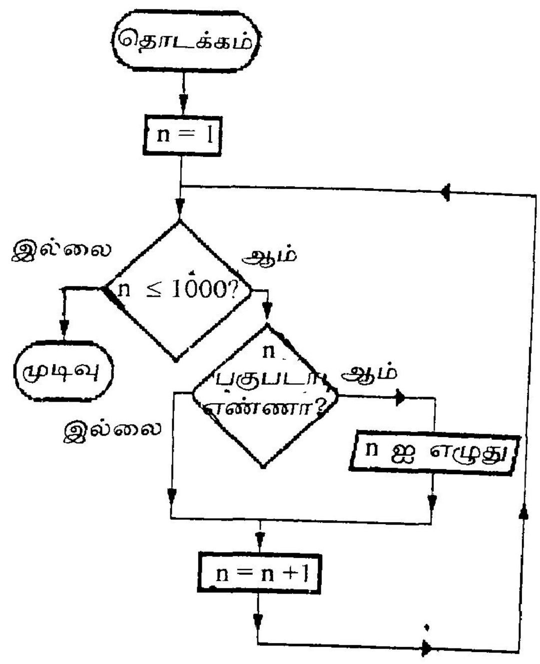
படம் 3 - 8 (5) நூறில் சிறியது



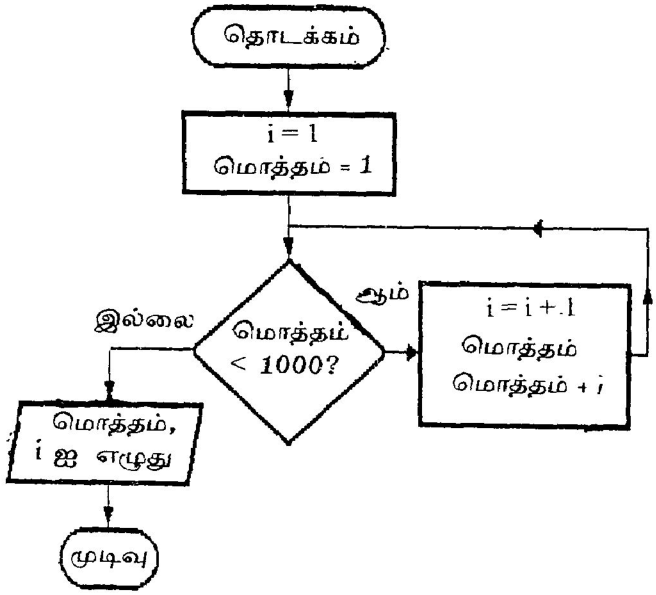
படம் 3 - 9 (6) வட்டிக் கணக்கு



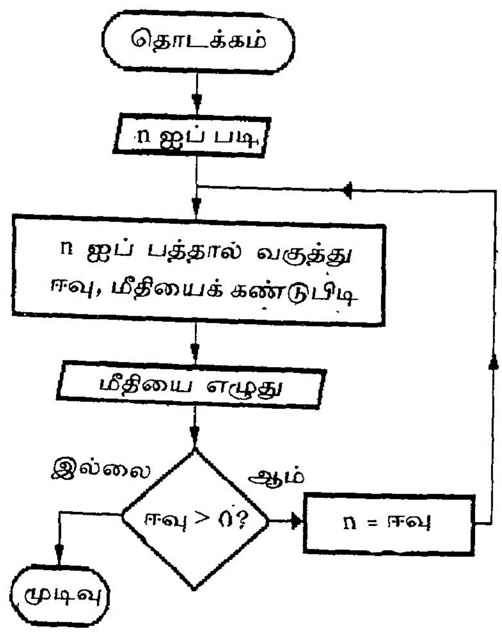
படம் 3 - 10 (7) பகுபடா எண்



படம் 3 - 11 (8) பல பகுபடா எண்கள்



படம் 3 - 12 (9) மொத்தம்



படம் 3 - 13 (10) இலக்கங்கள்

3.4 அடிப்படைக் கட்டுப்பாட்டு அமைப்புகள்

சென்ற பாடத்தில் பல எடுத்துக்காட்டுகளைக் கொடுத்ததற்கு இரண்டு முக்கிய காரணங்கள் உண்டு. இந்த வழிப்படங்களை எழுத முடிந்தால்தான் ஒருவரால் கட்டளை நிரல்களை எழுத முடியும். ஏனென்றால் ஒரு கட்டளை நிரலை எழுத, கணிப்பொறி செய்ய வேண்டிய செயல்கள் ஒவ்வொன்றையும் நாம்தான் கூற வேண்டும். அத்துடன், எந்தச் செயலுக்குப்பின் எதைச் செய்ய வேண்டும், எப்படி இருக்கும்போது எந்தச் செயலைச் செய்ய வேண்டும் என்ற நுணுக்கங்களையும் நாம்தான் கூறவேண்டும். நமக்கு இவை எல்லாம் தெரிந்தால்தான் கட்டளை நிரலை எழுத முடியும். ஓர் கணக்கிற்கு விடைகாண, கணிக்கும் முறையை நாம்தான் விவரமாகக் கூற வேண்டும். கணிப்பொறி அந்த விதத்தில் அப்படியே சிறிதும் மாறாமல் கணித்துக் கொடுக்கும்; தானாக எதுவும் செய்யாது. கணிப்பொறியைப் பரமார்த்த குருவின் ஐந்தாவது சீடன் என்று கூடக்கூறலாம்! எப்படி என்கிறீர்களா?

அழகான இந்த பரமார்த்த குரு கதைகளில் குதிரை வாங்கிவந்த கதையைப் படித்திருக்கிறீர்களா? (அவசியம் படித்துப் பார்த்து சிரித்து மகிழ வேண்டிய தமிழ்க் கதைகள் இவை.) அதை சுருக்கமாகப் பார்ப்போம். பரமார்த்த குருவுக்கு மட்டி, மடையன், மூடன், முட்டாள் என்று நான்கு சீடர்கள். பெயருக்கேற்ற அறிவுத்திறன் படைத்தவர்கள். குருவுக்கு ஒரு கிழட்டுக் குதிரை வாங்கி, குருவை அதில் உட்கார வைத்து அழைத்து வருகிறார்கள். வழியில் ஒரு மரக்கிளை தட்டிவிட குருவின் தலைப்பாகை கீழே விழுகிறது. சிறிது நேரம் கழித்து, குரு சீடர்களிடம் தலைப்பாகையைக் கேட்கிறார். அவர்கள் அது கீழே விழுந்து விட்டது என்கிறார்கள். ''கீழே விழுந்ததை எடுக்க வேண்டும் என்பது தெரியாதா?'' என்று குரு கேட்க, ''இனிமேல் அப்படியே செய்கிறோம்'' என்று சொல்லிவிட்டு, ஒரு சீடன் ஓடிப்போய் தலைப்பாகையை எடுத்து வருகிறான்! வரும்போது வழியில் குதிரை இட்ட சாணத்தையும் அதில் எடுத்து வருகிறான்! ஏன் அதை எடுத்து வந்தான் என்பதற்கு ''நீங்கள்தானே கீழே விழுந்ததை எடுக்க வேண்டும் என்று சொல்லியிருக்கிறீர்கள்'' என்று பதில் வருகிறது. சிலவற்றை மட்டும் எடுக்க வேண்டும், மற்றதை எடுக்கக் கூடாது என்று குரு கூற, தங்களுக்கு அது தெரியாது என்று கூறி, எடுக்க வேண்டிய பொருள்களின் பட்டியல் ஒன்றைக் குருவிடம் கேட்டு எழுதிவைத்துக் கொள்கிறார்கள்.

சிறிது நேரத்தில் கிழட்டுக் குதிரை தடுக்கி விழ, குரு கீழே விழுகிறார். வயது முதிர்ந்த குருவால் எழ முடியவில்லை. சீடன் ஒருவன் பட்டியலைப்படிக்க, தலைப்பாகை, துண்டு, வேட்டி என்று ஒவ்வொன்றாகக் குருவுடன் கீழே விழுந்த பொருள்களை எடுத்து விடுகிறார்கள். குரு கோவணத்துடன் கீழே கிடக்கிறார். தன்னைத் தூக்கிவிடும்படிச் சொல்கிறார். அதற்கு ஒரு சீடன் ''கீழே விழுந்த

எல்லாவற்றையும் எடுக்கக் கூடாது. பட்டியலில் உள்ளது மட்டும்தான் எடுக்கவேண்டும் என்று சொல்லியிருக்கிறீர்கள். நீங்கள் அந்தப் பட்டியலில் இல்லை'' என்று கூறுகிறான். அந்தப் பட்டியலில் என்னையும் சேர்த்துக் கொள்ளுங்கள் என்று குரு கூற, பட்டியல் மாற்றப்பட்டு பின்னர் குருவும் தூக்கி விடப்படுகிறார்.

அந்தப் பட்டியல்தான் கட்டளை நிரல். கணிப்பொறி இன்னும் ஒரு சீடன். அவ்வளவுதான். அதிக வித்தியாசமில்லை. ஆகவே கணக்கை நன்றாக ஆராய்ந்து, அதைத் தீர்ப்பதற்கான செயல் முறையை விளக்கமாக அளிப்பது நம் பொறுப்பு. இப்படி விளக்கமாகச் சிந்திப்பதில் பயிற்சி மிக அவசியம். பல வழிப்படங்களைக் கொடுத்ததற்கு இது ஒரு நோக்கம்.

அடுத்த நோக்கம், இந்த வழி முறைகளிலெல்லாம் பொதுவாகக் காணப்படும் மூன்று அடிப்படைச் செயல்பாடுகளை அறிந்து கொள்வது. இவற்றைத்தான் நாம் திரும்பத்திரும்ப எல்லாக் கணிப்புகளிலும் பயன்படுத்தப் போகிறோம்! ஒவ்வொரு கணிப்பையும் எப்படி இந்த அமைப்புகளை வைத்து எழுத வேண்டும் என்று சிந்தித்தால் எளிதில் அதற்கான செயல்முறை கிடைத்து விடும். இப்படிச் சிந்திப்பதுதான் சிறந்த வழி என்றும் கண்டுபிடித்திருக்கிறார்கள். அடிப்படை அமைப்புகளை அறிந்து கொள்வதும், அவற்றை அடிப்படையாகக் கொண்டு சிந்திப்பதும், கடற்கரையில் குடியிருக்கும் மீனவன் நீந்தக் கற்றுக் கொள்வதற்கு ஒப்பாகும். அவ்வளவு முக்கியம்.

வரிசை முறை

சாதாரணமாக, கணிப்பொறியானது கொடுக்கும் உத்தரவுகளை ஒவ்வொன்றாக வரிசையாகச் செயல்படுத்தும். இந்த **வரிசை முறை** (sequencing) ஓர் அடிப்படை அமைப்பு.

பிரிதல்

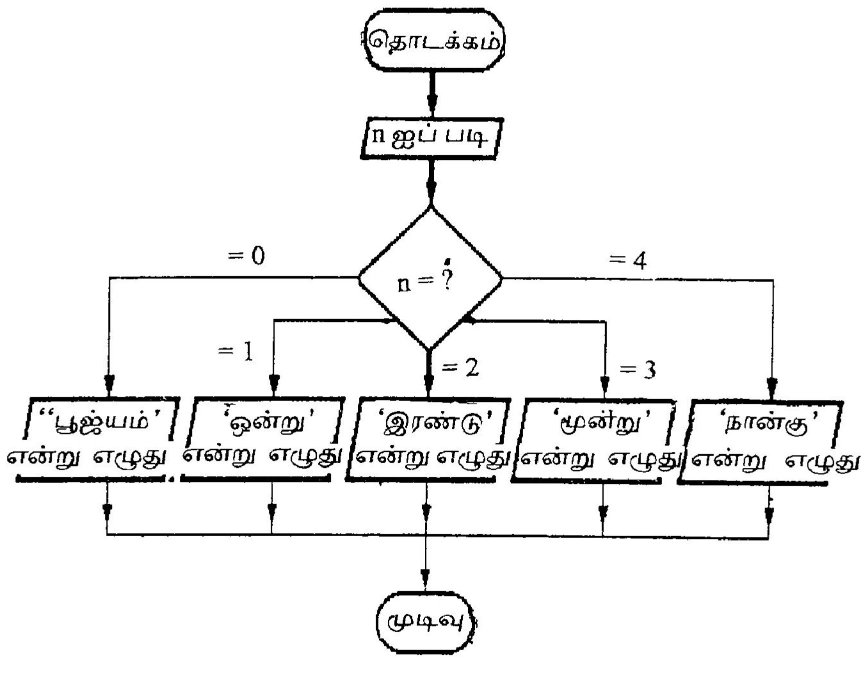
ஒரு கேள்வி கேட்டு, அதற்குக் கிடைக்கும் ஆம் / இல்லை என்ற விடையின் அடிப்படையில், பிரிந்து செல்லும் இரண்டு பாதைகளில் ஒன்றைத் தேர்ந்தெடுத்து, அந்த வழியில் செல்வதைக் குறிக்கும் தீர்மானக் குறியீடான **சாய்சதுரம்** (Diamond) இங்கு பல வழிப்படங்களிலும் இருப்பதைப் பார்க்கலாம். ஒரே படத்தில் பல முறையும் பார்க்கலாம். இவ்வாறு, அப்போதுள்ள சூழ்நிலையை வைத்து ஒரு கிளைப் பாதையைத் தேர்ந்தெடுத்து, அந்த வழியில் செல்வது, **பிரிதல்** (Branching) என்னும் அடிப்படை அமைப்பு.

உதாரணமாக, A > B என்றால் Aஐ அச்சிடு இல்லையென்றால் Bஐ அச்சிடு என்பதைக் கூறலாம். இதைப் பொதுவாக, ''.......என்றால்....என்று செய், இல்லையென்றால்...என்றுசெய்'', என்ற அமைப்பு எனலாம். தேர்ந்தெடுக்கப்பட வேண்டிய இரு பாதைகளில் ஒன்றில் எதுவும் செய்யத் தேவையில்லாமல்

இருக்கலாம். அந்த அமைப்பை, ...என்றால்... என்று செய்.., என்று குறிப்பிடலாம். இதில் 'இல்லையென்றால் எதுவும் செய்ய வேண்லாம்; அடுத்த உத்தரவுக்குப்போ' , என்பது சொல்லாமலேயே அடங்கும். இந்த இரண்டு அமைப்புகளும் ஆங்கிலத்தில் "if...then...else....", என்றும் "if...then...", என்றும் குறிப்பிடப்படும்.

இரு கிளைப் பாதைகளில் ஒன்றில் பிரிந்து செல்வதற்குப் பதிலாக, பல கிளைப் பாதைகள் இருந்து அவற்றில் ஒன்றைத் தேர்ந்தெடுத்து, அதில் பிரிந்து செல்ல வேண்டியிருக்கலாம். இதைப் படம் 3.5 இல் உள்ளது போல் பல இரு வழிப் பிரிவுகள் மூலம் செயல்படுத்தலாம். இதையே சற்று எளிமையாக்கி படம் 3.14 இல் உள்ளது போல் குறிப்பிடலாம். இதைப் **பலவழிப் பிரிதல்** (Multiway branching) என்று கூறலாம். '...என்பது ,..,...என்றால்...என்று செய்,... என்றால்..,என்று செய்' என்று குறிப்பிடலாம். உதாரணமாக,

n என்பது
0 என்றால் 'சுழி' என்று எழுது,
1 என்றால் 'ஒன்று' என்று எழுது.
2 என்றால் 'இரண்டு' என்று எழுது.
3 என்றால் 'மூன்று' என்று எழுது.
4 என்றால் 'நான்கு' என்று எழுது.



படம் **3 - 14** பலவழிப் பிரிதல்

பலமுறை செய்தல்

மூன்றாவது அடிப்படை அமைப்பு **'பலமுறை செய்தல்'** (Looping), ஒரே செயலை, தகவல்களை மட்டும் மாற்றி, திரும்பத் திரும்பச் செய்வது. உதாரணங்கள்,

1. புதிய புதிய எண்களை நூறுமுறை படிப்பது.
2. வட்டியை 20 முறை கணக்கிடுவது.

இந்த இரண்டு உதாரணங்களிலும், எத்தனை முறை செய்யப்போகிறோம் என்பதை முதலிலேயே தீர்மானம் செய்து விடுகிறோம். முதலில் (0 அல்லது) 1 என்று ஆரம்பித்து எத்தனை முறை இதுவரை செய்திருக்கிறோம் என்பதை கணக்கில் வைத்துக் கொள்கிறோம். இதற்கு i,j போன்ற பெயர்களை எண்ணுவதற்காகவே பயன்படுத்துகிறோம். இவை Index Variable எனப்படும்.

இந்த **எண்ணும் பெயரைப்** பயன்படுத்துவதில் நான்கு செயல்பாடுகள் அடங்கியிருக்கும். அவை

1. எண்ணும் பெயருக்கு ஒரு ஆரம்ப மதிப்பைக் கொடுக்க வேண்டும்.
2. இதன் மதிப்பை, இறுதி மதிப்புடன் ஒப்பிட்டுப் பார்த்து, இனி ஒருமுறை செய்ய வேண்டுமா என்று பார்க்க வேண்டும்.
3. ஆம் என்றால்,

   தேவையான செயல்களை ஒரு முறை செய்ய வேண்டும்.

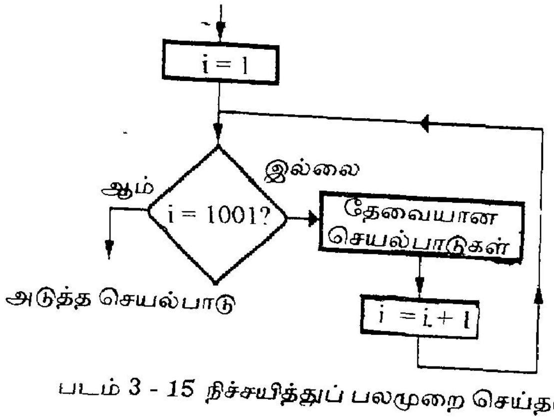
   எண்ணும் பெயரின் மதிப்பை 1 கூட்ட வேண்டும்.

   மீண்டும் 2 ஆம் பத்திக்குப் போக வேண்டும்.
4. இல்லை என்றால்,

   வேலை முடிந்துவிட்டது.

   அடுத்த உத்தரவுக்குப் போகலாம்.

இதற்கான வழிப்படத்தை, ஓர் உதாரணத்தை படம் 3.15 இல் காணலாம்.



படம் 3 - 15 நிச்சயித்துப் பலமுறை செய்தல்

எத்தனை முறை செய்யவேண்டும் என்பதை முதலிலேயே தீர்மானித்து விடுவதால் இதற்கு '***நிச்சயித்துப் பலமுறை செய்தல்***' (Defenite Iteration)*என்று பெயர். நாம் சென்ற பாடத்தில் பார்த்த உதாரணங்களில் 5 முதல் 8 வரை உள்ளவற்றில் இந்த அமைப்பு வருவதைப் பாருங்கள். ஒவ்வொன்றிலும் எண்ணும் பெயரையும், அது எப்படி மாறுகிறது, எப்படிச் செயல்படுகிறது என்பதையும் அறிந்து கொள்ளுங்கள்.*

நிச்சயமற்று பலமுறை செய்தல்

சென்ற பாடத்தில் பார்த்த ஒன்பதாம், மற்றும் பத்தாம் உதாரணங்களிலும் பலமுறைகள் செய்வதை, அம்புக்குறிகளால் ஏற்படும் ஒரு வளையம் காண்பிக்கிறது. ஆனால் இவற்றில் எண்ணும் பெயர் எதுவும் இல்லை. ஏனென்றால் எத்தனை முறை திரும்பச் செய்ய வேண்டும் என்று ஆரம்பத்தில் தெரியாது. எத்தனை எண்களை கூட்டினால் 1000 அல்லது அதற்கு மேல் வரும் என்று தெரியாது. அதனால், எத்தனை முறை கூட்ட வேண்டும் என்பது தெரியாது. ஓர் எண்ணில் எத்தனை இலக்கங்கள் இருக்கும் என்பது தெரியாது. அதனால், இந்த இடங்களில், இன்னும் ஒரு முறை செய்ய வேண்டுமா என்பதை வேறு ஏதாவது ஒரு நிபந்தனையைச் சரிபார்ப்பதன் மூலம் செய்ய வேண்டும். நிபந்தனை ஏற்கப்பட்டால், அதாவது ஆம் என்ற விடை கிடைத்தால் இன்னும் ஒரு முறை செய்ய வேண்டும். இல்லை என்றால் அடுத்த செயலுக்கு போய்விடலாம். '' மொத்தம் < 1000?'', '' *எண்* > 0?" *என்ற நிபந்தனைகளை நாம் சரிபார்க்கிறோம். இந்த முறை* ***நிச்சமற்றுப் பலமுறை செய்தல்*** (Indefenite Iteration) *எனப்படும்.*

வரிசையாகச் செய்தல், பிரிதல், பலமுறை செய்தல் என்ற மூன்று அடிப்படைக் கட்டுப்பாட்டு அமைப்புகளைக் கொண்டு பல கணிப்புகளையும் செய்துவிட முடியும். எந்த உத்தரவுக்குப் பிறகு எந்த உத்தரவைச் செயல்படுத்த வேண்டும் என்பதை இந்த அமைப்புகள் கட்டுப்படுத்துவதனால் இவை ***கட்டுப்பாட்டு அமைப்புகள்*** (Control Structures) *எனப்படும்.*

நினைவில் நிற்க

1. வரிசையாகச் செய்தல், பிரிதல், பலமுறை செய்தல் என்பவை அடிப்படைக் கட்டுப்பாட்டு அமைப்புகள்.

2. பிரிதலில், இருவழிப்பிரிதல், பலவழிப்பிரிதல் என இருவகை உண்டு.

3. பலமுறை செய்தலில், நிச்சயித்துப் பலமுறை செய்தல், நிச்சயமற்றுப் பலமுறை செய்தல் என இருவகை உண்டு.

4. முதல் வகையில் எத்தனை முறை செய்திருக்கிறோம் என்று கணக்கிட ஒரு எண்ணும் பெயர் தேவை.

5. இரண்டாம் வகையில் எத்தனை முறை செய்திருக்கிறோம் என்பது கணக்கிடப்படுவதில்லை. ஒவ்வொரு முறையும் ஒரு நிபந்தனை சரிபார்க்கப் படுகிறது.

பயிற்சி வினாக்கள்

1. அடிப்படைக் கட்டுப்பாட்டு அமைப்புகள் எவை?
2. பலவழிப்பிரிதலுக்கு ஓர் உதாரணம் கொடு. அதை எவ்வாறு இருவழிப் பிரிதல்களைக் கொண்டே செயல்படுத்த முடியும் என்று காண்பி.
3. பலமுறை செய்தலில் உள்ள இரண்டு வகைகளுக்கும் உள்ள வித்தியாசங்கள் யாவை?
4. நிச்சயித்துப் பலமுறை செய்தலில் எண்ணும் பெயரைப் பயன்படுத்தும் முறையை விவரி.
5. சென்ற பாடத்தில் எழுதிய வழிப்படங்களில் அடிப்படைக் கட்டுப்பாட்டு அமைப்புகள் மட்டுமே பயன்படுத்தப்பட்டுள்ளனவா என்று பார்.
6. சென்ற பாடத்தில் எழுதிய வழிப்படங்களில் உள்ள நிச்சயித்துப் பலமுறை செய்தல், நிச்சயமற்றுப் பலமுறை செய்தல் என்னும் அமைப்புகளை அடையாளம் காண். அவற்றில் பயன்படுத்தப்படும் எண்ணும் பெயர்களையும், நிபந்தனைகளையும் கண்டுபிடி.

3.5 போலிக் குறியீடுகள்

செயல்முறைகளை வழிப்படத்தைக் கொண்டு எழுதுவதைப்போல, **போலிக் குறியீடுகளைக்** கொண்டும் (Pseudo Code) எழுதலாம். போலிக் குறியீடு என்றால் என்ன? இது ஆங்கிலத்திற்கும், உயர்நிலைக் கணிப்பொறி மொழிக்கும் இடைப்பட்ட ஒரு நிலை. ஆங்கிலத்தில் வாக்கியங்கள் நீளமாக இருக்கும். உயர்நிலை மொழிகளில் இலக்கணக் கட்டுப்பாடுகளைச் சிறிதும் வழுவாமல் பின்பற்ற வேண்டும். இந்த இரண்டு தொந்தரவுகளையும் நீக்கிவிட்டால் கிடைப்பது போலிக்குறியீடு.

இதில் மிகச்சில வகை வாக்கியங்களே உண்டு. அவையும் சின்னச் சின்ன வாக்கியங்கள். கடினமான இலக்கணம் இல்லை. ஆனால் நாம் செய்ய வேண்டிய கணிப்பு எல்லாவற்றையும் சொல்லும் ஆற்றல் உண்டு. இன்னொரு சௌகரியமும் உண்டு. நமக்குத் தேவையான சில வாக்கியங்களை நாமே கூட உருவாக்கிக் கொள்ளலாம்.

கீழே உள்ள வகை வாக்கியங்களை மட்டுமே வைத்து செயல் முறைகளை எழுதலாம். அவை-

1. if....then...else...
2. if...then...
3. for...to...do...
4. while....do...

இவற்றைத் தமிழிலும் எழுதலாம், இவ்வாறு-

1. ...என்றால்...என்றுசெய், இல்லையென்றால்...என்றுசெய்.
2. ...என்றால்...என்றுசெய்.
3. ...என்பது...முதல்...வரை...என்றுசெய்.
4. ...என்றிருக்கும் வரை...என்று செய்.

இவற்றில் முதலிரண்டும் பிரிதலைக் குறிக்கின்றன. மூன்றாவது, நிச்சயித்துப் பலமுறை செய்தலையும், நான்காவது நிச்சயமற்றுப் பல முறை செய்தலையும் குறிக்கின்றன. இவற்றுக்குச் சில உதாரணங்களையும், அவற்றுக்கான வழிப்படங்களையும் காண்போம்.

1. if a > b then print a else print b.

   a < b என்றால் aஐ அச்சிடு இல்லையென்றால் bஐ அச்சிடு.

2. if a < 10 then b = c + d

   a < 10 என்றால் b = c + d என்றுசெய்.

3. for i = 1 to 20 do

n = n + i

i என்பது 1 முதல் 20 வரை

n = n + i

என்று செய்.

4. while sum < 1000 do (sum = Sum + i; i = i + 1)

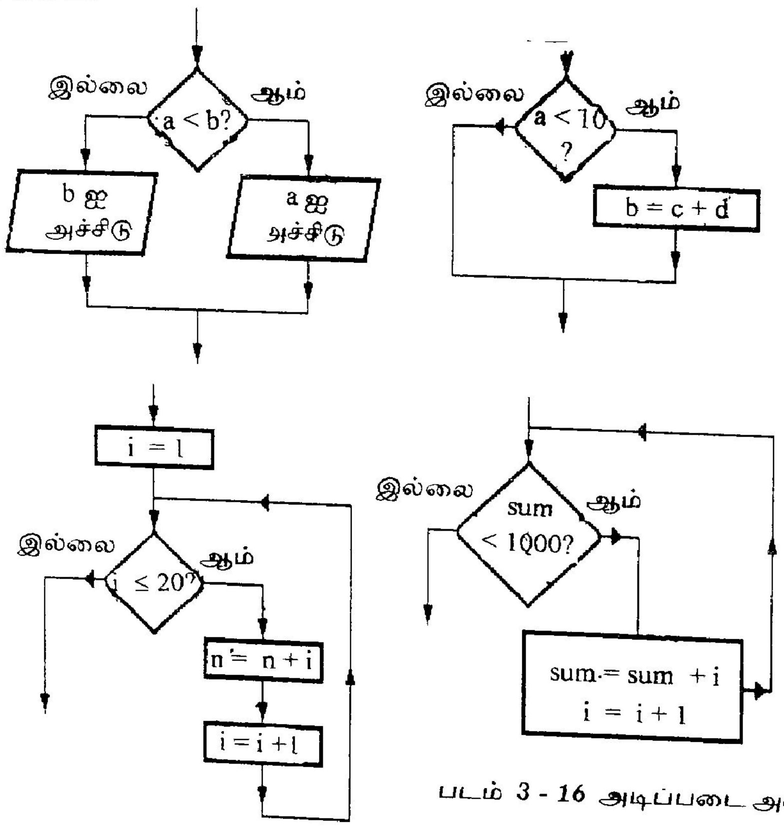
sum < 1000 என்றிருக்கும் வரை

sum = sum + i

i = i + 1

என்று செய்.

இவற்றுக்கான வழிப்படங்கள் படம் 3.16 இல் உள்ளன. இவற்றில் எல்லாமே உள்ளே செல்ல ஒரே வழியும், வெளியே வர ஒரே வழியும் இருப்பது இவற்றின் சிறப்பு, பலம்.



படம் 3 - 16 அடிப்படை அமைப்புகள்

நாம் ஏற்கனவே பார்த்த பத்துக் கணக்குகளுக்கு செயல்முறையைப் போலிக் குறியீட்டில் தமிழில் எழுதலாம். இவற்றை வழிப்படங்களுடன் ஒப்பிட்டுப்பார்.

1. கொள்ளளவு கண்டுபிடிப்பது.

1. நீளம், அகலம், உயரம் இவற்றைப் படி.

2. கொள்ளளவு = நீளம் x அகலம் x உயரம்.

3. கொள்ளளவு என்பதை அச்சிடு.

4. முற்றும்.

இதில் வரிசை முறை மட்டுமே பயன்படுத்தப்பட்டிருக்கிறது.

2. எழுத்தால் எழுதுவது.

1. n ஐப் படி

2. n என்பது

(0 என்றால் 'சுழி' என்று எழுது.

1 என்றால் 'ஒன்று' என்று எழுது.

2 என்றால் 'இரண்டு'என்று எழுது.

3 என்றால் 'மூன்று' என்று எழுது.

4 என்றால் 'நான்கு' என்று எழுது.)

3. முற்றும்

எளிமை கருதி ...என்பது...என்றால்...என்று செய் என்ற குறியீடு பயன்படுத்தப்பட்டிருக்கிறது. இதையே இப்படியும் எழுதலாம்.

1. n ஐப்படி

2. n = 0 என்றால் 'சுழி' என்று எழுது.

3. n = 1 என்றால் 'ஒன்று' என்று எழுது.

4. n = 2 என்றால் 'இரண்டு' என்று எழுது.

5. n = 3 என்றால் 'மூன்று' என்று எழுது.

6. n = 4 என்றால் 'நான்கு' என்று எழுது.

7. முற்றும்.

*இங்கு பல என்றால் ...என்று செய் என்னும் அமைப்புகள் ஒன்றுக்கு அடுத்து ஒன்றாக, **வரிசையாக** எழுதப்பட்டுள்ளன. ஆனால் வழிப்படத்தில் என்றால்...என்றுசெய், இல்லையென்றால்.... என்ற அமைப்புகள் **ஒன்றுக்குள் ஒன்றாகக்** கொடுக்கப்பட்டிருக்கின்றன. இரண்டுமே சரிதான். ஆனால், வித்தியாசமானவை என்பதைத் தெரிந்துகொள்ள வேண்டும்.*

3. சிறிய எண்

1. a, b, c, இவற்றைப்படி

2. a < b என்றால்

(a < c என்றால் aஐ வெளியிடு.

இல்லையென்றால் cஐ வெளியிடு.

)

இல்லையென்றால்

(b < c என்றால் bஐ வெளியிடு.

இல்லையென்றால் bஐ வெளியிடு.

)

3. முற்றும்

இங்கு கவனிக்க வேண்டிய முக்கிய விஷயங்கள் மூன்று.

1. ஓர் என்றால்...இல்லையென்றால் வாக்கியத்திற்குள் இன்னொரு என்றால்...இல்லையென்றால் வாக்கியம் இருக்கிறது. ஒரு வாக்கியத்திற்குள் இன்னொரு வாக்கியம் இருக்கும்போது, அதைத் தெளிவாகக் காட்ட, உள் வாக்கியத்தைச் சற்று இடம்விட்டு எழுதுகிறோம்.

2. கணித இயலில் அடைப்புக் குறிகளைப் பயன்படுத்தி பலவற்றை ஒன்று சேர்த்துக் காண்பிப்பது போல், இங்கேயும் அடைப்புக் குறிகளைப் பயன்படுத்யிருக்கிறோம். இது பல செயல்களைச் சேர்த்து ஒரே செயல்போல் காண்பிப்பதற்காக. இதனால் செயல்பாடுகளை எழுதுவதும், புரிந்துகொள்வதும் எளிதாகின்றன.

3. உள்வாக்கியங்கள் மட்டுமே சற்று இடம் விட்டு எழுதப்படுகின்றன. ஆனால், ஒரு வாக்கியத்தின் அடுத்த வாக்கியம், சரியாக அதற்குக் கீழே வர வேண்டும். உதாரணமாக இங்கு 1,2,3 என்று வரும் வாக்கியங்கள் ஒரே இடத்தில் ஆரம்பிப்பதைக் கவனி.

4. இருபடிச் சமன்பாடு

1. a, b, c இவற்றைப்படி.

2. (a = 0) என்றால்

('இது இருபடிச் சமன்பாடு இல்லை' என்று எழுது. முற்றும்)

இல்லையென்றால்

((d = b^2 - 4ac.

(d < 0) என்றால்

('விடைகள் இமாஜினரி எண்கள்' என்று எழுது. முற்றும்)

இல்லையென்றால்

((d = 0) என்றால்

(r = - b/a.

'இரு சமமான விடைகள்' என்று எழுது.

r, r ஐ எழுது.

)

இல்லையென்றால்

(e = d.

r1 = (- b + e) / 2a.

r2 = (- b - e) / 2a

'இரு விடைகள்' என்று எழுது.

r1, r2 ஐ எழுது.

)

)

3. முற்றும்

5. நூறு எண்களில் சிறியது

1. aஐப் படி.
2. min = a.
3. எண்ணிக்கை என்பது 2 முதல் 100 வரை

(a ஐப் படி,

(a < min) என்றால் min = a.

)

4. min ஐ அச்சிடு.
5. முற்றும்.

6. வட்டிக் கணக்கு

1. முதல், வட்டி விகிதம் - இவற்றைப்படி.
2. வருடம் என்பது 1 முதல் 20 வரை

(வட்டி = முதல் X வட்டி விகிதம் / 100.

முதல் = முதல் + வட்டி.

வருடம், முதல் - இவற்றை எழுது.

3. முற்றும்.

7. பகுபடா எண்

1. n - ஐப் படி.
2. i என்பது 2 முதல் n - 1 வரை
    (i என்பது n ஐ மீதியில்லாமல் வகுக்கும்) என்றால்
        (n வகுபடும் எண் என்று எழுது. முற்றும்)
)
3. n வகுபடா எண் என்று எழுது
4. முற்றும்.

8. பல பகுபடா எண்கள்

1. n என்பது 1 முதல் 1000 வரை
((n பகுபடா எண்) என்றால்
n ஐ வெளியிடு.
)
2. முற்றும்.

9. மொத்தம்

1. i = 1.
2. மொத்தம் = 1.
3. (மொத்தம் < 1000) என்றிருக்கும் வரை
    (i = i + 1.
    மொத்தம் = மொத்தம் + i
)
4. மொத்தம், i இவற்றை எழுது.
5. முற்றும்.

10. இலக்கங்கள்

1. n ஐப்படி.
2. nஐ 10 ஆல் வகுத்து ஈவு, மீதியைக் கண்டுபிடி.
3. மீதியை எழுது.
4. (ஈவு > 0) என்றிருக்கும்வரை
    (n = ஈவு.
    nஐ 10 ஆல் வகுத்து ஈவு, மீதியைக் கண்டுபிடி.
    மீதியை எழுது
)
5. முற்றும்.

கொடுத்த எண் 0 என்றால், 0 என்பதை எழுத வேண்டும். அதற்காகத்தான் முதலில் ஒரு முறை வரிகள் 2, 3 ஐ தனியாகச் செய்கிறோம். பிறகு திரும்பச் செய்யும் வளையத்திற்குள் போகிறோம்.

வழிப்படத்தையும், போலிக் குறியீட்டையும் ஒப்பிட்டால்,

1. வழிப்படத்தைவிட போலிக்குறியீட்டில் எழுதுவதும், எழுதியதைப் புரிந்து கொள்வதும் எளிது.

2. நிச்சயித்துப் பலமுறை செய்தலில், எண்ணிக்கையைக் கணக்கு வைத்துக் கொள்ளும் விவரங்களை வழிப்படத்தில் நாமே கொடுக்க வேண்டும். போலிக் குறியீட்டில் எழுதப்படும் செயல்முறைகள், கிட்டத்தட்ட கணிப்பொறி மொழியில் எழுதப்பட்டவை போல் இருப்பதால், இதிலிருந்து கட்டளை நிரல்களை எழுதுவது மிகவும் சுலபம்.

நினைவில் நிற்க

1. மிகச்சில போலிக்குறியீடுகளை வைத்து எல்லாக் கணிப்புகளையும் செய்யலாம். இவை ஓரளவு கணிப்பொறி மொழி போல் இருப்பதால், இவற்றிலிருந்து கட்டளை நிரல்கள் எழுதுவது எளிது.

பயிற்சி வினாக்கள்

1. முக்கிய போலிக் குறியீடுகளைப் பட்டியலிடு. ஒவ்வொன்றுக்கும் இரண்டு உதாரணங்கள் கொடு. இவற்றின் செயல்பாட்டை வழிப்படங்கள் மூலம் காண்பி.

3.6 உலாவருதல்

ஒரு கணிப்பைச் செய்ய, ஒரு செயல் முறையானது இயற்கை மொழியில் அல்லது வழிப்படத்தில் அல்லது போலிக்குறியீட்டில் உருவாக்கப்பட்டு, அதிலிருந்து உயர்நிலை மொழியில் ஒரு கட்டளை நிரல் எழுதப்படுகிறது. இந்தக் கட்டளை நிரல் நமக்குத் தேவையான விடைகளைச் சரியாகக் கணித்துக் கொடுக்குமா என்பது மிக முக்கியமான கேள்வி. அதனால் செயல் முறையை, வழிப்படத்தை, போலிக்குறியீட்டை, கட்டளை நிரலை சரியாக இருக்கிறதா என்று பார்க்க வேண்டியது அவசியமாகிறது (சரியான கட்டளை நிரல்களை எழுதுவது மாணவர்களுக்கும் அவசியம். மதிப்பெண்கள் பெறவேண்டுமே!)

முதலில் **செயல்முறை** (Algorithm) என்பது எப்படி இருக்க வேண்டும் என்று பார்ப்போம். அடுத்து, வழிப்படம், போலிக்குறியீடு மற்றும் கட்டளை நிரல்களை ஓரளவு சரிபார்க்க உதவும் **உலாவருதல்** (Walk through) என்னும் உத்தியைப் பார்ப்போம்.

சரியான செயல்முறை கீழே உள்ள விதிமுறைகளுக்குக் கட்டுப்பட்டு இருக்க வேண்டும்.

1. செயல் முறையில் உள்ள பாகங்கள் அல்லது படிகள் வரைநிலைக்கு உட்பட்ட எண்ணிக்கையில் (Finite number of steps) இருக்க வேண்டும்.

2. ஒவ்வொரு படியும் குழப்பத்திற்கு இடமில்லாமல் செயலாற்றுமாறு இருக்க வேண்டும் (Unambiguous).

3. ஒவ்வொரு படியும் குறிப்பிட்ட அளவு நினைவகத்தைப் பயன்படுத்தி, குறிப்பிட்ட காலத்திற்குள் செய்து முடிக்கப்படுவதாக இருக்க வேண்டும் (Finite space and finite time).

4. மொத்தச் செயல்பாடும் வரைநிலைக்கு உட்பட்ட காலத்திற்குள் செய்து முடிக்கப்படுவதாக இருக்க வேண்டும் (Finite time).

இப்படிப்பட்ட செயல்பாடு ஒன்று வழிப்படமாக நம்மிடம் தரப்பட்டிருக்கிறது எனலாம். கணிப்பொறியானது இந்த வழிப்படத்தை வைத்துக் கொண்டு செயல்பட்டால், அது எப்படிச் செயல்படும் என்று பார்ப்பதுதான் உலாவருவதின் நோக்கம். நாம்தான் கணிப்பொறி என்று நினைத்துச் செயல்பட வேண்டும். தொடக்கம் என்ற இடத்தில் ஆரம்பிப்போம். வழிப்படத்தில் தொடக்கம் என்பது ஒரே ஒரு இடத்தில்தான் இருக்கும். (ஆனால் 'முற்றும்' பல இடங்களில் இருக்கலாம்) இங்கிருந்து செல்லும் அம்புக்குறி வழியாக அடுத்தபெட்டிக்குச் செல்வோம். அது ஒரு செயல்பெட்டியாக இருந்தால் அந்தச் செயலைச் செய்வோம். உள்ளீடு செய்யும் பெட்டியாக இருந்தால், எந்தெந்தப் பெயர்களுக்கு

எந்தெந்தப் மதிப்புகள் அளிக்கிறோம் (உள்ளீடு செய்கிறோம்) என்பதைக் குறித்துக்கொள்ள வேண்டும். வெளியீட்டுப் பெட்டியாக இருந்தால், குறிப்பிடப்பட்டுள்ள பெயர்களின் தற்போதைய மதிப்புகளை வெளியீடு செய்வதாக நினைக்க வேண்டும். தீர்மானக் குறியீடான சாய்சதுரம் என்றால், தேவையான கேள்விக்கு ஆம் அல்லது இல்லை என்ற விடையைக் கண்டுபிடித்து, இரு கிளைப்பாதைகளில் ஒன்றைத் தேர்ந்தெடுத்து, அந்த வழியில் உள்ள அடுத்த பெட்டிக்குச் செல்ல வேண்டும்.

இப்படி ஒவ்வொரு பெட்டியாகக் கடந்து சென்று, முடிவு என்னும் பெட்டியை அடையும்போது ஓர் உலா முடிவடைகிறது. இந்த உலாவில், உள்ளீடு செய்த மதிப்புகளுக்கு ஏற்ப கணிப்பு நடைபெறுகிறது. கிடைக்கும் விடை சரியானதா என்று பார்க்க வேண்டும். இல்லை என்றால், தவறு எங்கே நடந்தது என்று மீண்டும் ஒரு முறை அதே வழியில் வந்து பார்க்க வேண்டும். ஓர் உலாவில் சரியான விடை கிடைத்துவிட்டது என்போம். இந்த உலாவில் நாம் சரிபார்த்தது தொடக்கத்திலிருந்து முடிவு வரை உள்ள ஒரு வழியில். இதேபோல் பல வழிகள் இருக்கலாம். உள்ளீடு செய்யும் தகவல்களின் மதிப்புகளைப் பொறுத்து நாம் செல்லும் வழிகளும் மாறுபடும். ஒவ்வொரு உலாவிலும் வெவ்வேறான மதிப்புகளை உள்ளீடு செய்து, எல்லா வழிகளிலும் கணிப்பு சரியாக நடைபெறுகிறது என்று கண்டறிந்தால், கொடுக்கப்பட்ட வழிப்படம் சரியானது என்று ஓரளவு கூறலாம். இதேபோல் போலிக் குறியீடு மற்றும் கட்டளை நிரல்களையும் ஓரளவு சரி பார்க்கலாம்.

நினைவில் நிற்க

1. ஒரு சரியான செயல் முறையானது - வரை நிலைக்கு உட்பட்ட எண்ணிக்கையில் படிகள்; குழப்பமற்று, குறிப்பிட்ட காலத்திற்குள் குறிப்பிட்ட நினைவகத்தைப் பயன்படுத்தி செயல்படும் படிகள்; மொத்தச் செயல்பாடுகளுக்கும் வரை நிலைக்கு உட்பட்ட நேரம் - என்ற விதிகளுக்கு உட்பட்டு இருக்க வேண்டும்.

2.உலாவருவதன் மூலம் தொடக்கத்திலிருந்து முடிவுவரை வரும் பல பாதைகள் சரியானபடி செயல்படுகின்றனவா என்று பார்க்கலாம்.

3. உள்ளீடு செய்யும் மதிப்புகளுக்கு ஏற்றாற்போல் பாதைகளும் மாறும்.

பயிற்சி வினாக்கள்

1. சென்ற பாடங்களில் பார்த்த 10 வழிப்படங்களிலும், 10 போலிக்குறியீடுகளிலும் உள்ள பல பாதைகளிலும் தகுந்த உள்ளீடுகளைத் தயார் செய்வதன் மூலம் உலா வருக.
   இது வழிப்படங்களையும், போலிக் குறியீடுகளையும் புரிந்து கொள்ள மிகவும் உதவும்.

பேசிக் மொழி - அறிமுகம்

4

பேசிக் மொழியில் சிறு கட்டளை நிரல்களை எழுதத் தேவைப்படும் அடிப்படைச் செய்திகளை இந்தப் பகுதியில் பார்ப்போம். எந்தக் கணக்கைத் தீர்ப்பதற்கும், அதற்கான தகவல்களைக் கணிப்பொறிக்கு உள்ளே செலுத்த வேண்டும். அதாவது உள்ளீடு செய்ய வேண்டும். அந்தத் தகவல்களை அலசி ஆராய்ந்து, கணித்து முடித்தவுடன் விடைகளை வெளியீடு செய்ய வேண்டும். தகவல்களை எவ்வாறு உள்ளீடு, வெளியீடு செய்வது என்பதை இப்பகுதியில் பார்ப்போம்.

தகவல்களை மாறி, மாறிலி என இரு வகைகளாகப் பார்க்கலாம். மேலும், எண்கள், சரங்கள் எனவும் பிரிக்கலாம். எண்களைக் கணிதச் செயல்பாடுகள் மூலம் கோர்த்து, புது எண்களை உருவாக்கலாம். செயல்பாடுகள் பலவற்றையும் நாம் ஏற்கனவே அறிந்துள்ள அடிப்படைக் கட்டுப்பாட்டு அமைப்புகள் மூலம் வரிசைப்படுத்தி, விடைகளைக் கணக்கிடலாம். இவ்வாறு சிறு கட்டளை நிரல்களை உருவாக்குவதை இப்பகுதியில் காண்போம்.

கணிப்பொறியின் முன் உட்கார்ந்து, கட்டளை நிரல்களைச் செயல்படுத்தி விடை காண்பதற்கு, கணிப்பொறிக்குச் சில ஆணைகள் இட்டு அதனிடம் வேலை வாங்கவேண்டும். இந்த ஆணைகள் சிலவற்றையும் இங்கு காண்போம்.

4.1 அறிமுகம்

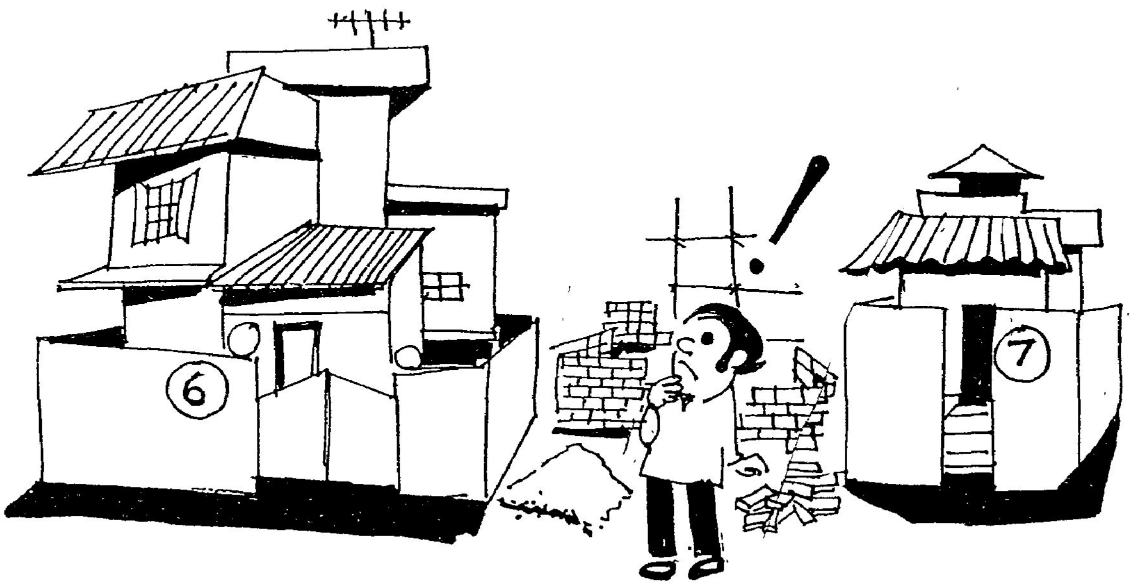
சித்திரமும் கைப்பழக்கம், செந்தமிழும் நாப்பழக்கம் - என்பது போலவே, கணிப்பொறி மொழி ஒன்றினைக் கற்றுத் தேர்ச்சி பெற, அம்மொழியில் பல கட்டளை நிரல்களை எழுதி, செயல்படுத்திப் பார்க்க வேண்டியது அவசியம். இதற்கு கணிப்பொறி இன்றியமையாதது. இப்புத்தகத்தைப் படிக்கும் சிலருக்கு கணிப்பொறி கிடைக்காத நிலை இருக்கலாம். அவர்களுக்கும் பயன்பட வேண்டும் என்ற நோக்கத்தில் இப்பகுதி சற்று விளக்கமாகத் தரப்பட்டுள்ளது. மேலும் இப்பகுதியில் கற்பவை மற்ற பல கணிப்பொறி மொழிகளுக்கும் ஓரளவு பொருந்தும்.

பேசிக் மொழிமாற்றிகள் பல உண்டு. GWBASIC, BASICA என்பவை மைக்ரோ சாஃப்ட் நிறுவனம் வெளியிட்டவை. இவற்றிலும் பல பதிப்புகள் உண்டு. ஒவ்வொன்றுக்கும் சிறிது வேறுபாடு இருக்கும். புதிய பதிப்புகளில் சில புதிய கட்டளைகள் சேர்க்கப்பட்டிருக்கும் அல்லது சில மாற்றப்பட்டிருக்கும். உதாரணமாக, புதிய பதிப்புகளில் மாறியின் பெயர் இரண்டு எழுத்துகளுக்கு மேல் இருக்கலாம். IF...THEN...ELSE மற்றும் WHILE வாக்கியங்களும் உண்டு. இங்கு கொடுக்கப்பட்டுள்ள பெரும்பாலான செய்திகள் மைக்ரோ சாஃப்ட் நிறுவனம் Personal Computer Hardware Reference Library என்னும் வரிசையில் வெளியிட்டுள்ள BASIC என்ற புத்தகத்தின் அடிப்படையில் அமைந்தவை. நுண்கணிப்பொறிகளில் பயன்படும் பேசிக் Micro computer (Enhanced) BASIC எனப்படும். பேசிக்குறியீடுகளிலிருந்து இந்த மேம்படுத்தப்பட்ட (Enhanced) பேசிக்கில் கட்டளை நிரலை எழுதுவது எளிது. அதனால் நாம் மேம்படுத்தப்பட்ட பேசிக்கையே பயன்படுத்துவதாக வைத்துக் கொள்வோம். அதனால் இரண்டு எழுத்துகளுக்கு மேற்பட்ட பெயர்கள், IF...THEN...ELSE, WHILE முதலியவற்றை நாம் பயன்படுத்துவோம்.

நாம் ஏற்கனவே பார்த்தது போல், நமக்குத் தேவையானவற்றைக் கணிக்க மூன்று நிகழ்ச்சிகள் நடைபெற வேண்டும். முதலாவதாக, கணிப்பிற்குத் தேவையான தகவல்களை கணிப்பொறிக்கு உள்ளீடு செய்ய வேண்டும். இரண்டாவதாக, இத் தகவல்கள் அலசப்பட்டு விடை கணக்கிடப்பட வேண்டும். மூன்றாவதாக, இவ்விடையைக் கணிப்பொறி வெளியீடு செய்ய வேண்டும். வெளியீடு செய்வது பலமுறைகளில் இருக்கலாம். திரையில் தெரியலாம், அச்சிட்டுக் கொடுக்கலாம், காந்த நாடாவில் பதியலாம் அல்லது காந்தத் தகட்டில் எழுதலாம். இதேபோல் தகவல்களைப் பல விதங்களிலும் உள்ளீடு செய்யலாம். விசைப்பலகை மூலம் அனுப்பலாம், காந்த நாடா, காந்தத் தகட்டிலிருந்து எடுக்கலாம், அல்லது கட்டளை நிரலிலேயே தகவலையும் பதித்து வைக்கலாம்.

தகவல்களை எந்தக் கருவி மூலம் உள்ளீடு, வெளியீடு செய்கிறோம் என்பது அவ்வளவு முக்கியமில்லை. அதனால், இதைப்பற்றித் தெரிந்துகொள்ள இப்போது நேரத்தைச் செலவிடாமல் மேலே செல்வோம். உள்ளீடு செய்வது விசைப்பலகை மூலம், அல்லது கட்டளை நிரலில் பதித்து வைத்து என்றும், வெளியீடு செய்வது திரை மூலம் என்றும் வைத்துக்கொள்ளலாம்.

கட்டளை நிரல் (Program) என்பது, ஒன்றன்பின் ஒன்றாக, வரிசையாக எழுதப்படும் கட்டளைகளின் தொகுப்பு. இப்படி வரிசையாக எழுதப்படு கட்டளைகள் சாதாரணமாக ஒரு வரிக்கு ஒரு கட்டளையாக இருக்கும். ஒரு தெருவில் வரிசையாக உள்ள வீடுகளை, சரியாக அடையாளம் காண, கதவு எண்கள் கொடுக்கப்படுவது போல், கட்டளைகளுக்கும் அடையாள எண்கள் உண்டு. இந்த எண்கள் பெரும்பாலும் 1, 2, 3,... என்று வரிசையாகக் கொடுக்கப்படாமல், 10, 20, 30, ... என்று கொடுக்கப்படும். இந்த **வரி எண்கள்** (Line numbers) எப்போதும் ஏறு வரிசையிலேயே அடுக்கி வைக்கப்படும். இவை 0 முதல் 65529 வரை இருக்கலாம். ஓர் எண்ணுக்கும் அடுத்த எண்ணுக்கும் இடையே நிறைய இடைவெளி இருக்கலாம். எதற்காக இப்படி இடைவெளி விட வேண்டும்? காரணமாகத்தான்.



"என் வீட்டுக்கு எந்த எண் கொடுப்பது?"

எந்தக் கட்டளை நிரலும் முதல் முறை எழுதும்போது முழுவதும் சரியானதாக இருக்காது. எழுத மறந்து போன அல்லது தேவையானசில கட்டளைகளை நடுவில் சேர்க்க வேண்டியிருக்கலாம். எண்கள் 40க்கும் 50க்கும் இடையில் ஒரு

கட்டளையைச் சேர்ப்பது எளிது. 41 முதல் 49 வரை உள்ள எண்களில் ஒன்றை எடுத்து,அந்த எண் உள்ள ஒரு வரியை எழுதவேண்டும். அவ்வளவுதான். நாம் பயன்படுத்தும் பேசிக் அமைப்பு(**மொழி மாற்றி** (Interpreter), **சொல் தொகுப்பான்** (Text Editor), **பிழை நீக்கி**(Debugger) மூன்றும் சேர்ந்த தொகுதி) தானாக இந்த வரியை 40க்கும் 50க்கும் இடையில் சேர்த்துவிடும். முதலில் எண்கள் கொடுக்கும்போது 1,2,3,...என்று கொடுத்திருந்தால்,பல கட்டளைகளின் எண்களை மாற்ற வேண்டுயிருக்கும். இந்தத் தொல்லையைத் தவிர்க்கத்தான் இடைவெளி விட்டுக் கட்டளை எண்கள் கொடுக்கப்படும். ஓர் எண்ணுக்கும் அடுத்த எண்ணுக்கும் இடைவெளி சமமாக இருக்க வேண்டியதில்லை. 10, 50, 75, 100, 110, 115 என்று கூடக் கொடுக்கலாம். ஏறுவரிசையில் இருக்க வேண்டியது மட்டும் அவசியம். பேசிக் மொழியில் கட்டளை எண்கள் கொடுக்கப்பட வேண்டியது அவசியம். மற்ற மொழிகளில் அப்படியில்லை.

கட்டளை நிரல்களில் சில இடங்களில் நமக்குப் புரிவதற்காக சில விளக்கங்களை எழுதி வைப்பது அவசியம். அப்போதுதான் அந்த நிரலை நாமோ அல்லது பிறரோ வேறொரு சமயத்தில் படிக்கும்போது நன்றாகப் புரியும். இந்த மாதிரியான வரிகளை, கணிப்பொறி எதுவும் செய்யாமல் விட்டுவிட்டு அடுத்தவரிக்குச் சென்றுவிடும். இம்மாதிரிக் **குறிப்பு வரிகளை** (Remarks) REM என்று ஆரம்பிக்க வேண்டும். இரு வரிகளுக்கு இடையில் இடம் விட்டு எழுதவும் இந்த குறிப்பு வரி பயன்படும். சில உதாரணங்கள்.

```
10 REM END OF CALCULATIONS
20 REM
40 REM PROGRAM FOR CLASS MARKS
```

இனி வரும் பாடங்களில் பல இடங்களில், ''இதைக் கணிப்பொறியே செய்து விடும்'' என்பது போன்ற வாக்கியங்கள் வரும். இதற்குத் தேவையான கட்டளை களை, பேசிக் மொழிமாற்றியே கொடுத்து, கணிப்பொறியைச் செய்யச் சொல்லிவிடும் என்பதுதான் இதன் பொருள்.

பேசிக்மொழியில், முழு எண்கள் தேவைப்படும் இடங்களில் எல்லாம் பின்னங்களையும் கொடுக்கலாம். சில மொழிமாற்றிகள் 2.8 என்பதை 2 என்று (பின்னப்பகுதியை முழுவதும் விட்டுவிட்டு) எடுத்துக் கொள்ளும். சில மொழி மாற்றிகள் உதாரணமாக BASICA இதை அருகிலுள்ள முழு எண்ணாக (இங்கு 3) எடுத்துக் கொள்ளும். நீங்கள் பயன்படுத்தும் மொழிமாற்றி எப்படிச் செய்யும் என்பதை சோதித்துப் பாருங்கள்.

சாதாரணமாக பேசிக் மொழியில், ஒரு வரிக்கு ஒரு வாக்கியம்தான் எழுதப்படும். மேம்படுத்தப்பட்ட பேசிக்கில் ஒரே வரியில் பல வாக்கியங்களை எழுதலாம். இரண்டு வாக்கியங்களுக்கு இடையில் முக்காற்புள்ளியான் (Colon) என்பதை இடவேண்டும். உதாரணம் -

```
10 A = 3: B = 7: N$ = "MANY"
```

நினைவில் நிற்க

1.பேசிக் மொழியில் ஒவ்வொரு வரிக்கும் ஓர் எண் கொடுக்கப்பட வேண்டும். ஒரே வரியில் பல கட்டளைகளைக் கூட எழுதலாம்.

2. இவை ஏறுவரிசையில் இருக்க வேண்டும். இடைவெளிவிட்டு இருப்பது நல்லது.

3. கணிப்பொறிக்கு இல்லாமல் படிப்பவர்களுக்கு உதவும் குறிப்புகளை எழுத REM வாக்கியம் பயன்படும்.

பயிற்சி வினாக்கள்

1. வரி எண்களை இடைவெளி விட்டு எழுதுவதன் நோக்கம் என்ன?
2. REM வாக்கியத்தின் பயன்பாடுகள் இரண்டினைக் கூறு.

4.2 வெளியீடு

எந்தக் கணிப்பிலும் உள்ளீடு, கணிப்பு, வெளியீடு என்ற மூன்று அங்கங்கள் உண்டு. இவற்றுள் எது இல்லையென்றாலும் கணிப்பு முழுமையாகாது. முதலில் வெளியீடு செய்வது எவ்வாறு என்பதை இங்கு பார்ப்போம். மற்ற பகுதிகளை எளிமையாக வைத்துக் கொள்ள, தேவையான தகவல்களை, கட்டளை நிரலுக்குள்ளேயே வைத்துக்கொள்வோம்.

கணிப்பொறி ''கண்ணன்'' என்ற பெயரை, திரையில் எழுத வேண்டும் என்போம். இதற்கான கட்டளை

10 அச்சிடு ''கண்ணன்''

பேசிக் மொழி ஆங்கிலத்தை அடிப்படையாகக் கொண்டது. அதனால் இதை

10 PRINT "KANNAN"

என்றுதான் எழுத வேண்டும். நமக்குத் தெரிந்த மிகச்சில ஆங்கில வார்த்தைகள் மட்டுமே இங்கு பயன்படுவதால் ஆங்கிலத்தில் எழுதுவது கடினமாக இருக்கப் போவதில்லை. PRINT *என்ற கட்டளையில், மேற்கோள் குறிக்குள் எதைக் கொடுத்தாலும், அது அப்படியே திரையில் எழுதப்பட்டுவிடும்.*

திரையைப் பற்றி சிறிது அறிந்து கொள்வது அவசியம். சாதாரணமாக, திரையில் ஒருவரியில் 80 குறியீடுகள் தெரியும். 24 வரிகள் தெரியும். பேசிக் மொழியில் PRINT *கட்டளைக்காக, ஒரு வரியை 5 சம பாகங்களாகப் பிரித்திருக்கிறார்கள். ஒவ்வொரு பாகமும் 16 குறியீடுகள் நீளம் கொண்டது; 1 முதல் 16 வரை, 17 முதல் 32 வரை, 33 முதல் 48 வரை, 49 முதல் 64 வரை, 65 முதல் 80 வரை. இப்போது கட்டளையை சிறிது மாற்றி எழுதுவோம்.*

10 PRINT "KANNAN", "KALA" ; "MALA"

திரையில் கண்ணன், கலா, மாலா என்ற மூன்று பெயர்களும் எழுதப்படும் என்பது புரிகிறது. இந்தப் பெயர்கள் எங்கே தெரியும்? ஒரே வரியிலா? பல வரிகளிலா? பெயர்களுக்கு நடுவில் இடம் விட்டா? இடம் விடாமலா? ஏன் எல்லாப் பெயர்களும் ஒரே ஜோடி மேற்கோள் குறிக்குள் இல்லாமல் தனித்தனியாக இருக்கின்றன? ஏன் ஓர் இடத்தில் காற்புள்ளியும், ஓர் இடத்தில் அரைப்புள்ளியும் இருக்கின்றன? இதனால் ஏதாவது மாற்றம் விளையுமா? இந்தக் கேள்விகளுக் கெல்லாம் விடை காணவேண்டும்.

ஒரு ஜோடி மேற்கோள் குறிக்குள் வருவது ஒரு ***சரம்*** (String of characters) *குறியீடுகளின் சரம் என்று எடுத்துக்கொள்ளப்படும். இங்கு மூன்று சரங்கள் உள்ளன. முதல் சரம் ஒரு வரியின் முதல் பாகத்தில் எழுதப்படும். இந்தப் பாகத்தில் உள்ள 16 இடங்களில்* KANNAN *என்ற சரத்தில் உள்ள 6 குறியீடுகள் முதல் 6 இடங்களை*

நிரப்பும். இப்போது ஒரு காற்புள்ளி வருகிறது. கணிப்பொறி இந்த காற்புள்ளியைப் பார்த்ததும், இந்தப் பாகத்தில் உள்ள மீதி இடங்களில் எதுவும் எழுதாமல் விட்டுவிட்டு, அடுத்த பாகத்திற்குச் சென்றுவிடும். இந்தப் பகுதியின் ஆரம்பத்தில் நான்கு இடங்களில் (17 முதல் 20 வரை) KALA என்று நான்கு எழுத்துகளை எழுதும். இப்போது கணிப்பொறி அரைப்புள்ளியைப் பார்க்கும். அதனால், விட்ட இடத்திலிருந்தே தொடர்ந்து எழுதத் தயாராகும். அடுத்தது MALA என்பதை 21 முதல் 24 வரை உள்ள இடங்களில் எழுதும். அதனால் திரையில் இப்படித் தெரியும்.

KANNAN ƀƀƀƀƀƀƀƀƀƀ KALAMALA

ƀ என்ற குறியீடு (Blank) ஒரு இடைவெளி என்பதைக் குறிக்கும். எத்தனை இடைவெளிகள் என்று சரியாகத் தெரிவதற்காக இந்தக் குறியீடு பயன்படுத்தப்படுகிறது. இதைப் பாருங்கள்.

```
200 PRINT "A", "B", "C", "D", "E", "F", "G"
```

காற்புள்ளிகள் இருப்பதால் ஒவ்வொரு எழுத்தும் ஒரு பகுதியில் வரும். அந்தப் பகுதியின் ஆரம்பத்தில் வரும். முதல் வரியில் உள்ள 5 பாகங்களில் E வரை வந்துவிடும். இப்போது இந்த வரி முடிவடைந்து விடுவதால், அடுத்த பாகம் அடுத்த வரியின் முதல் பாகமாக வரும். அதனால் F மற்றும் G அடுத்த வரியில் உள்ள முதல் இரண்டு பாகங்களில் வரும். திரையில் இவ்வாறு தெரியும்.

```
A    B    C    D    E
F    G
```

மேற்கோள் குறிக்குள் கொடுக்கப்படும் சரங்கள் அப்படியே திரையில் வரும்; சாவிப்பலகையில் உள்ள இடைவெளிச் சாவியைத் தட்டி உண்டாக்கும் இடைவெளிகள் உட்பட,

```
70 PRINT "GOOD MORNING"
```

என்ற கட்டளை, GOOD ƀ MORNING என்ற சரத்தை எழுதும்.

ஒரு கட்டளை மூலம் பல வரிகள் எழுத முடியும் என்று பார்த்தோம். பல கட்டளைகள் இருந்தால் ஒரே வரியில் எழுத முடியுமா? முடியும். ஒரு PRINT கட்டளையின் முடிவில் காற்புள்ளி அல்லது அரைப்புள்ளி இல்லாவிட்டால், அந்தக் கட்டளைக்கான வெளியீடு முடிந்ததும், கணிப்பொறியானது அடுத்த வரிக்குப் போய்விடும், புதிதாக வருவதை வெளியீடு செய்ய. அதனால்,

```
10 PRINT "THIS"
20 PRINT "IS"
30 PRINT "INDIA"
```

என்ற கட்டளைகள்

```
THIS
IS
INDIA
```

என்று வெளியீடு செய்யும்.

ஒரு PRINT *கட்டளையின் முடிவில் காற்புள்ளி அல்லது அரைப்புள்ளி இருந்தால், பிறகு வருபவற்றை வெளியீடு செய்ய அதே வரியில் காத்திருக்கும். காற்புள்ளியானால் அதே வரியில் எழுதி முடித்த பகுதிக்கு அடுத்த பகுதியிலும், அரைப்புள்ளியானால் அப்போது எழுதி முடித்த இடத்திலும் காத்துநிற்கும். அந்த வரியில் எல்லாப் பகுதிகளும் முடிந்துவிட்டால் அல்லது 80 இடங்களும் நிரப்பப்பட்டுவிட்டால், வேறு வழியில்லாமல் அடுத்த வரிக்குப் போகும். இந்த முறையில் பல* PRINT *கட்டளை மூலம் ஒரே வரியில் எழுத முடியும். எடுத்துக்காட்டாக,*

```
10 PRINT "A", "B";
20 PRINT "C", "D";
30 PRINT "E", "F",
40 PRINT "G"
```

என்பது ABCƀƀƀƀƀƀƀƀƀƀƀƀƀ DEFƀƀƀƀƀƀƀƀƀƀƀƀƀ G *என்று வெளியீடு செய்யும்.*

ஒரு பகுதியில் எதுவும் எழுதாமலேயே, அந்தப் பகுதியைக் கடந்துசெல்ல முடியும், எடுத்துக்காட்டாக,

```
10 PRINT , "GOOD" , "BOY"
```

என்பதைப் பார்ப்போம். கணிப்பொறி ஆரம்பத்தில் முதல் பகுதியில் எழுதக் காத்திருக்கும். முதல் காற்புள்ளியைப் பார்த்து இரண்டாம் பகுதிக்குப் (17 ஆம் இடத்திற்கு) போகும். அடுத்த காற்புள்ளியைப் பார்த்து மூன்றாம் பகுதிக்குப் போகும். அங்கு GOOD *என்று எழுதியதும், ஒரு காற்புள்ளியைப் பார்த்து நான்காம் பகுதிக்கும், அடுத்த காற்புள்ளியைப் பார்த்து ஐந்தாம் பகுதிக்கும் போகும். அங்கு* BOY *என்று எழுதும்.*

ஒரு வரியில் எதுவும் எழுதாமலேயே அடுத்த வரிக்குப் போகலாம். அதற்கு, என்ன எழுதவேண்டும் என்பதைக் கூறாமல் விட்டுவிட வேண்டும். அதாவது

```
10 PRINT "A"
20 PRINT
30 PRINT "B"
```

என்ற தொடர், முதல் வரியில் A *என்றும், மூன்றாவது வரியில்* B *என்றும் எழுதும்.*

நினைவில் நிற்க

1. PRINT கட்டளை வெளியீடு செய்ய உதவும்.

2. காற்புள்ளி, அரைப்புள்ளிகள் தகவல்களை அழகாக வெளியிட உதவும்.

3. காற்புள்ளியானது அடுத்த பகுதிக்குப் போ என்பதையும், அரைப்புள்ளி யானது இங்கேயே தொடர்ந்து எழுது என்பதையும் குறிக்கும்.

4. இவை கட்டளையின் இறுதியில் வந்தால், பின்னால் வெளியீடு செய்வதை அதே வரியில் மேலே சொன்னது போல் எழுதும்.

பயிற்சி வினாக்கள்

1. திரையில் ஒரு வரியில் சாதாரணமாக எத்தனை எழுத்துகள் தெரியும்? அவை எத்தனை பகுதிகளாகப் பிரிக்கப்பட்டிருக்கும்?
2. PRINT வாக்கியத்தில் காற்புள்ளியும், அரைப்புள்ளியும் பயன்படும் விதத்தை எடுத்துக்காட்டுகளுடன் விளக்கு.
3. முதல் வரியில் முதல் பாகத்தில் COME என்றும் நான்காம் வரியில் நான்காம் பாகத்தில் HERE என்றும் எழுதக் கட்டளைகள் எழுது.
4. கீழ்க்காணும் கட்டளைத் தொடர்கள் எவ்வாறு வெளியீடு செய்யும் என்பதனைக் காண்பி. இடைவெளிகளை சரியாகக் காண்பி.

அ.
```
10 PRINT "MAX" ; "GOOD", "BAD",
20 PRINT "GET" ; "MIX"
30 PRINT "FIX"
```

ஆ.
```
100 PRINT, "RAJ" , , "RASANNA"
150 PRINT, "MAN" ; "TEN",
160 PRINT, "GET"
```

இ.
```
10 PRINT, , "FIRST"
20 PRINT
30 PRINT "CLASS"
```

ஈ.
```
10 PRINT "GET" ; "100"
20 PRINT
30 PRINT
40 PRINT "BALLS"
```

உ.
```
100 PRINT "GOOD MORNING"
200 PRINT "ƀ RAJAN"
```

5. கீழ்க்காணுமாறு வெளியீடு செய்ய, தேவையான கட்டளைத் தொடர்களை எழுது.

அ.
```
I AM A GOOD BOY
YES
YES YES
```

ஆ.
```
LET       ME        SING
A         S         0
N         G
```

இ.
```
LET       ME        SING      AND       DANCE
NOW
```

ஈ.
```
A = 25   B = 4   C = 5  XX = 10
```

உ.
```
SMALL
BIG
```

6. கீழ்க்காணும் கட்டளைத் தொடர்களில் தவறுகள் எவையேனும் இருந்தால் கூறு :

அ. 10 PRINT "GET",

ஆ. 10 PRINT "S"

இ. 100 PRINT, "X" "Y"

ஈ. 101 PRINT "G", "O", "O"; "D"

உ. 107 PRINT "PRINT A + B"

ஊ. 10 PRINT 25

4.3 மாறிகள், மாறிலிகள்

சென்ற பாடத்தில் வெளியீடு செய்யவேண்டிய தகவல்களை எளிமை யானவையாக எடுத்துக்கொண்டோம். அவை எல்லாமே குறியீடுகளின் சரங்களாக (Character strings) இருந்தன. ஒவ்வொரு சரமும் வெவ்வேறு மதிப்புள்ளவை. கண்ணன் என்பதும் கலா என்பதும் வெவ்வேறுதானே. இந்தச் சரங்களின் மதிப்பு மாறாதவை. கண்ணன் என்றுமே கண்ணன்தான். ஆகவே இந்தச் சரங்கள் **சரமாறிலிகள்** (String Constants) எனப்படும்.

சரங்களில் மாறிகளும் உண்டு. முதலில் சில எடுத்துக் காட்டுகளைப் பார்ப்போம்.

1. இந்தப் பள்ளியின் தலைமை ஆசிரியர் பெயர் திரு. அன்பழகன்.

2. இந்த வீட்டில் குடியிருப்பவரின் பெயர் திரு. கண்ணன்.

3. இன்றைக்குத் தொலைக்காட்சியில் செய்தி வாசித்தவர் திருமதி. மேனகா. இந்தப் பெயர்கள் எல்லாம் ஒரு குறிப்பிட்ட சமயத்தில் சரியாக இருக்கலாம்; காலம் மாற மாற இந்தப் பெயர்கள் மாறும். தலைமை ஆசிரியரின் அறை வாசலில் இருக்கும் பெயர்ப் பலகையில் அவ்வப்போது பெயர் மாறும். அதேபோல் குடியிருக்கும் வீட்டு வாசலில் உள்ள பெயர்ப்பலகையிலும் பெயர்கள் மாறிக் கொண்டே இருக்கும். பெயர்ப் பலகை ஒன்றுதான். ஆனால் அதில் எழுதி வைக்கப்படும் பெயர்கள் மாறிக்கொண்டே இருக்கும். தலைமை ஆசிரியரின் பெயர் என்பது பல நேரங்களில் பல **மதிப்புகள்** (Values) பெறும். இந்த மதிப்புகள் ஒவ்வொன்றும் ஒரு சர மாறிலி. ஆனால் தலைமை ஆசிரியரின் பெயர் என்பது ஒரு **மாறி** (Variable). இந்த மாறி சரங்களை மதிப்பாகப் பெறுவதால், **சர மாறி** (String Variable) எனவும் அழைக்கப்படும். இந்த வீட்டில் குடியிருப்பவர், இன்று செய்தி வாசித்தவர், பயணம் செய்யும் கார் இவை எல்லாமே சரமாறிகள்தான்.

குறியீடுகளின் சரங்களைப் பற்றிப் பேசிக் கொண்டிருக்கிறோம். என்னென்ன குறியீடுகளைப் பயன்படுத்தலாம் என்று தெரிந்துகொள்ள வேண்டாமா? குறியீடு களை மூன்று வகைகளாகப் பிரிக்கலாம். 0 முதல் 9 வரை உள்ள **இலக்கங்கள்** (Digits), ஆங்கில **எழுத்துகள்** (A முதல் Z வரை, a முதல் z வரை), மற்றும் +, -, *, / , =, $, ;, ,, , ", (,), >, < போன்ற **சிறப்புக்குறியீடுகள்** (Special symbols). ஒரு சரத்தில் அதிகபட்சம் எத்தனை குறியீடுகள் இருக்கலாம் என்பது நாம் பயன்படுத்தும் பேசிக் அமைப்பைப் பொறுத்தது. சாதாரணமாக 256 குறியீடுகள் இருக்கலாம். பேசிக்மொழியில் எப்போதும் A முதல் Z வரை பெரிய (Capital) எழுத்துகள்தான் பயன்படுத்தப்படும். சர மாறிலிகளில் மட்டும் சிறிய எழுத்துகளும் (a முதல் z வரை) பயன்படுத்தப்படும்.

சரங்களின் பயன்பாட்டைவிட, எண்களின் பயன்பாடு அதிகம். எண்களிலும், மாறிகளும், மாறிலிகளும் உண்டு. 67, 6.3, 10,027 போன்றவை மாறிலிகள். இன்றைய தேதி, என் மாதச் சம்பளம், கையிலிருக்கும் புத்தகங்களின் எண்ணிக்கை, பயணச்சீட்டின் விலை, இன்றைய விற்பனை போன்றவை மாறிகள். இவற்றின் மதிப்பு நேரத்திற்கு ஏற்ப மாறுபடும். **எண் மாறிலிகளை** number அல்லது numeric constant என்றும், **எண் மாறிகளை** Variable அல்லது Numeric Variable என்றும் கூறுவர்.

எண்களை பேசிக் மொழியில் எழுதும்போது கவனிக்க வேண்டிய சில விதிமுறைகள் -

1. இலக்கங்களுக்கு இடையில் காற்புள்ளி வைக்கக்கூடாது. 10,027 என எழுதக்கூடாது. 10027 என்றுதான் எழுத வேண்டும்.

2. எண்ணுக்கு முன்னால் +, - குறி வரலாம்.

3. எண்களை அடுக்குக் குறியீட்டில் (Exponential notation) எழுதலாம். அதாவது 10027 என்பது 0.10027×10^5. இதை அடுக்குக் குறியீட்டில் 0.10027E5 என்று எழுதலாம். இதையே 0.10027E + 5 அல்லது 1.0027E4 அல்லது 100.27E2 என்றும் எழுதலாம். 0.0012 என்பதை 0.12E-2 அல்லது 1.2E-3 என்று எழுதலாம்.

4. எண்களில் மிகப்பெரியதாகவும், மிகச் சிறியதாகவும் எவற்றைப் பயன்படுத்தலாம் என்பது நம் பேசிக் மொழிமாற்றியைப் பொறுத்தது.

5. எண்களில் ஒரு முறை தசமப்புள்ளி வரலாம்.

எண்மாறிகளைக் குறிக்க மேலே நீளமான பெயர்களைப் பயன்படுத்தினோம் (உதாரணமாக, என் மாதச் சம்பளம்). பேசிக்மொழியில், ஆரம்ப காலத்தில், எண் மாறியைக் குறிக்கும் பெயர்கள் ஓர் எழுத்தாகவோ, அல்லது, ஓர் எழுத்தும், அதற்கடுத்து ஓர் இலக்கமுமாகவோ இருக்க வேண்டும். உதாரணமாக X, A2, A9, M, N2 போன்றவை. இப்போது வரும் மொழிமாற்றிகள் பல எழுத்துகளை அனுமதிக்கும். பல ஆங்கில எழுத்துகளும் பல இலக்கங்களும் இருக்கலாம். ஆனால் முதல் குறியீடு ஒரு ஆங்கில எழுத்தாக இருக்க வேண்டும். உதாரணமாக, X, BAG, MAXIMUM, BIG24, SALARYX92, F27G4 போன்றவை அனுமதிக்கப் பட்டவை.

சர மாறிகளைக் குறிப்பிடும் பெயர்களின் இறுதியில் ஒரு டாலர் குறியீடு சேர்க்கப்பட வேண்டும். உதாரணம் X$, M2$, NAME$, CITY$, HEAD$, STATE$

இந்த மாறிகளை கணிப்பொறி எவ்வாறு பயன்படுத்துகிறது? ஓர் எண் மாறியை, கட்டளை நிரலில் குறிப்பிட்டால், கணிப்பொறியின் நினைவகத்தில் ஓர் எண் கொள்ளுமளவுக்கு இடத்தைத் தேர்ந்தெடுக்கும். இந்த மாறிக்கு இந்த இடம் என்பதையும் கணிப்பொறி நினைவில் வைத்துக்கொள்ளும். கட்டளை நிரலில் அந்த

மாறிக்கு ஒரு மதிப்பைக் கொடுத்தால், அந்த மாறியின் இடத்தில் அந்த மதிப்பை எழுதிவைக்கும். தேவையானபோது அந்த மாறியின் மதிப்பை எடுத்துப் பயன்படுத்தும். ஒரு மாறியின் மதிப்பை, தேவையானபோதெல்லாம் மாற்றி மாற்றி எழுதலாம். ஒரு மாறிக்கு எப்படி மதிப்பளிப்பது என்பதைப் பார்ப்போம்.

```
10 LET A = 44
20 LET B = A + 2
30 PRINT A, B
40 LET A = 20
50 PRINT A, B
60 LET B = A + B
70 PRINT A, B
```

LET என்பது **மதிப்பளிக்கும் வாக்கியம்** (Assignment statement) எனப்படும். LET A = 44 என்றால், ''A என்ற மாறியின் மதிப்பை 44 என்று வைத்துக்கொள்'' என்று பொருள். உடனே கணிப்பொறி Aக்காக ஒதுக்கப்பட்ட இடத்தில் 44 என்று எழுதிவைக்கும். அடுத்து வரும், LET B = A + 2 என்பதற்கு, ''தற்போதைய Aஇன் மதிப்புடன் இரண்டைக் கூட்டி, அதை Bஇன் மதிப்பாக வை'' என்பது பொருள்.

இங்கு = என்ற குறியீட்டை சமன்பாட்டுக் குறியீடாக நாம் பயன்படுத்த வில்லை. மதிப்பளிக்கும் செயல்பாட்டுக்கான குறியீடாகப் பயன்படுத்துகிறோம். அதனால் LET A = A + 5 என்று எழுதினால் தவறில்லை. இது Aஇன் தற்போதைய மதிப்புடன் ஐந்தைக் கூட்டி, புதிய மதிப்பாக வைக்கும். ஒவ்வொரு முறையும் LET என்று எழுதுவதற்குப் பொறுமை இல்லையா? பரவாயில்லை. மதிப்பளிக்கும் வாக்கியத்தை LET இல்லாமலேயே எழுதலாம். உதாரணமாக A = A + B என்பதும் சரிதான்.

எண்மாறிகளைப் போலவே சர மாறிகளையும் பயன்படுத்தலாம். உதாரணங்கள்:-

```
10 LET NAME$  =  "RAINBOW"
20 LET OLD$   =  NAME$
30 LET NAME$  =  "GOLD"
```

இந்தக் கட்டளைத்தொடர் செயல்பட்டால், முதலில் NAME$ என்னும் மாறி RAINBOW என்ற மதிப்பைப் பெறும். அடுத்து, OLD$ என்பதும் RAINBOW என்ற மதிப்பைப் பெறும். கடைசியாக NAME என்பது GOLD என்ற புதிய மதிப்பைப் பெறும்.

நாம் ஏற்கனவே எண்மாறிகளைக் கொண்டு பார்த்த உதாரணத்தில், A மற்றும் Bஏன் மதிப்புகள் எவ்வாறு மாறுகின்றன என்பதை இந்தப் பட்டியல் விளக்கும்.

வரி எண்	:	10	30	50	70
A	:	44	44	20	20
B	:	-	46	46	66

அதனால், அந்த கட்டளைத்தொடரில் உள்ள வெளியீடுகள் இப்படி இருக்கும்.

```
44      46
20      46
20      66
```

எண்களை PRINT *வக்கியத்தின் மூலம் வெளியிடும்போது கவனிக்க வேண்டியவை இரண்டு. எண்களுக்கு இடையில் அரைப்புள்ளி பயன்படுத்தும் போது, ஒவ்வொரு எண்ணுக்கு வலது புறமும் ஓர் இடைவெளி வரும். எதிர்மறை எண் என்றால், எண்ணுக்கு இடதுபுறம் ஒரு கழித்தல் குறியும், நேர்மறை எண் என்றால் எண்ணுக்கு இடதுபுறம் ஓர் இடைவெளியும் வரும். உதாரணமாக,* A = 22.33, B = -7.6 *என்றால்,*

PRINT A; B; A *என்பது* b̸ 22.33 b̸ - 7.6 b̸b̸ 2.33 *என வெளியீடு செய்யும். இதனால் எண்கள் ஒன்றுடன் ஒன்று ஒட்டாமல் வெளிவரும்.*

நினைவில் நிற்க

1. மாறிகளில் எண்மாறி, சரமாறி என இருவகைகள் உண்டு.

2. மாறிகள் சிறு சிலேட்டுகள் போல செயல்படும். அழித்து அழித்து எழுதலாம்.

3. சரமாறிகளின் பெயர்களின் இறுதியில் $ குறியீடுவரும்.

4.மாறிகளின் பெயர்களின் முதல் குறியீடு ஓர் எழுத்தாக இருக்க வேண்டும்.

5. மதிப்பளிக்கும் செயல்பாடு ஓர் அடிப்படைச் செயல்பாடு.

பயிற்சி வினாக்கள்

1. மாறிகளின் பெயர்களைப் பற்றிய விதிமுறைகள் யாவை? உதாரணத்துடன் விளக்கு.

2. கீழ்க்காணும் தொடர்களில் தவறு ஏதேனும் இருந்தால் கூறு.

அ. LET A = "WELL"

ஆ.LET A = WELL

இ. LET A$ = 46

ஈ. 93 = A3

உ. SALIRY = 9300

ஊ. A33 = A34

எ. A = B$ + 72.3

ஏ. LET X$ = 66 + 43

ஐ. LET BOMBAY$ = "BOMBAY"

ஒ. LET A = A + A + A

4.4. எண்மானச் செயல்பாடுகள்

எண்களை வைத்துக் கணக்கிடும்போது, கூட்டல், கழித்தல், பெருக்கல், வகுத்தல் மற்றும் அடுக்குதல் (Exponentiation) முதலிய எண்மானச் செயல்பாடுகளைச் (Numerical Operations) செய்கிறோம். இவை முறையே +, -, *, / , ↑, என்ற குறியீடுகளால் குறிப்பிடப்படுகின்றன. எடுத்துக்காட்டு -2 + 3, (2 * 3)-4.3, (2/3)↑4. 2^3 என்பதை 2↑3 என்றும், $\frac{2}{3}$ என்பதை 2/3 என்றும் எழுத வேண்டும். கணிப்பொறியில் ஒரு வரியில் மேலும் கீழும் எழுதுவது கடினம். அதனால்தான் இந்த ஏற்பாடு.

எண் மாறிலிகளுடன் எண் மாறிகளையும் எண்மானச் செயல்பாடுகளையும் பயன்படுத்தி **எண்மானக் கோவைகளை** (Numerical Expression) உருவாக்கலாம். உதாரணம் : A + (2.3 * SALARY) - TAX + 200. எண்மானக் கோவைகளை மதிப்பிட்டால் ஓர் எண் கிடைக்கும். மாறிகளைக் கோவைகளில் பயன்படுத்தினால், அந்த மாறிகளுக்கு அப்போது உள்ள மதிப்பைப் பயன்படுத்தி அந்தக் கோவையை மதிப்பிட வேண்டும். X என்பதன் தற்போதைய மதிப்பு 25 என்றால், (X * 2) + 10 என்ற கோவையின் மதிப்பு (25 x 2) + 10 = 60.

கோவைகளில் நமக்குத் தேவையானபடி அடைப்புக் குறிகளைப் பயன் படுத்திக்கொள்ளலாம். அடைப்புக் குறிகளைப் பயன்படுத்தினால், எந்தச் செயல்பாடு முதலில் செய்யப்பட வேண்டும் என்ற குழப்பத்திற்கு இடம் இருக்காது. அடைப்புக்குறிகள் இல்லாமலேயும் கோவைகளை எழுதலாம். (ஆனால் அப்படி எழுதுவதைத் தவிர்க்க வேண்டும். நல்லதில்லை. தவறுகள் நேர வாய்ப்பு உண்டு.) எடுத்துக்காட்டாக

X - A + 3/B * 7 ↑ 2 என்று எழுதலாம். செயல்பாடுகளை எந்த வரிசையில் செய்வது என்பதற்கு கணிப்பொறி ஒரு விதிமுறையைக் கையாள்கிறது. எண்மானச் செயல்பாடுகளுக்கான முன்னுரிமையை (Heirarchy of Operations) மூன்று நிலைகளாகப் பிரிக்கிறது.

1. அடுக்குதல் - முதல் நிலை.
2. பெருக்கலும் வகுத்தலும் - இரண்டாம் நிலை.
3. கூட்டலும் கழித்தலும் - மூன்றாம் நிலை.

ஒரே நிலையிலுள்ள செயல்களுக்குள் முன்னுரிமை எதுவுமில்லை; இடமிருந்த வலமாக வரும்போது ஒரே நிலையிலுள்ள குறிகளில் எது முதலாவதாக வருகிறதோ அதைச் செய்யும்.

கணிப்பொறி செயல்படும் விதத்தை எளிமையாக்கி இப்படிக் கூறலாம்.

1. இடமிருந்து வலப்புறம் செல். எங்கெங்கு அடுக்குதல் வருகின்றனவோ அவற்றையெல்லாம் செய்து கொண்டே செல்.

2. திரும்பவும் இடமிருந்து ஆரம்பித்து வலப்புறம் செல். எங்கெங்கு பெருக்கல் அல்லது வகுத்தல் வந்தாலும் அவற்றை செய்து கொண்டே செல்.

3. மூன்றாம் முறையாக இடமிருந்து வலப்புறம் செல். கூட்டல், கழித்தல் எது வந்தாலும் செய்து கொண்டே செல்.

மேலே பார்த்த எடுத்துக்காட்டை மீண்டும் பார்ப்போம்.

X - A + 3 / B * 7 2

இதில் முதல் முறை இடமிருந்து வலம் செல்லும் போது 7↑2 என்பது முதலில் கணக்கிடப்படும். அடுத்த முறை வரும்போது, முதலில் 3 / B கணக்கிடப்படும்; அடுத்து (3 / B) * (7↑ 2) என்பது கணக்கிடப்படும். இங்கு (3 / B) என்பதும் (7↑2) என்பதும் ஏற்கனவே கணக்கிடப்பட்ட இரு எண்கள். மூன்றாம்முறை இடமிருந்து வரும்போது, முதலில் X - A, அதற்கடுத்து + இவற்றைச் செய்யும், அதாவது, X - A + 3 / B * 7 - 2 என்பது

(X - A) + ((3 / B) * (7-2)) என்பதற்குச் சமம்.

இதை மதிப்பிடும்போது X = 4, A = 5, B = 3 என்று இருப்பதாக வைத்துக் கொள்வோம். செயல்பாடுகள் எந்த வரிசையில் செயல்பட்டு விடை எப்படி உருவாகும் என்பதைக் கீழே உள்ள படம் விளக்கும்.

*4 - 5 + 3 / 3 * 7↑2*

49

1

49

- 1

48

இதைப் போலவே கீழே உள்ள சமன்பாடுகளைச் சரிபார்க்கலாம்.

A / B * C / D = ((A / B) * C) / D

A↑B * C↑D = (A↑B) * (C↑D)

A / B / C = (A / B) / C

A/ (B / C) என்பதை ஒரே வரியில் கணக்கிட வேண்டுமென்றால் கண்டிப்பாக அடைப்புக்குறிகள் தேவை.

நினைவில் நிற்க

1. எண் மாறிகள், எண் மாறிலிகள், எண்மானச் செயல்பாடுகள் இவற்றைக் கொண்டு எண்மானக் கோவைகளை எழுதலாம்.

2. எண்மானக் கோவையை மதிப்பிட்டால் ஓர் எண் கிடைக்கும்.

3. அடைப்புக் குறிகளைத் தேவையான இடங்களில் பயன்படுத்தலாம்.

4. அடைப்புக்குறிகள் இல்லாத இடங்களில், செயல்பாடுகளின் முன்னுரிமைப்படி, செயல்பாடுகள் செயல்படுத்தப்படும்.

பயிற்சி வினாக்கள்

1. எண்மானச் செயல்பாடுகளின் முன்னுரிமையை விளக்கு. உதாரணங்கள் கொடு.

2. கீழே உள்ள எண்மானக் கோவைகளின் மதிப்பைக் கண்டுபிடி. எண்மானச் செயல்பாடுகளை வரிசைப்படுத்திக் காண்பி. A = 5, B = 4.2, C = 7, D = 8, E = 2 என இருக்கட்டும்.

அ. A + 3 * 4 + 12 / 6 ஆ. A * D / E + 4

இ. A - B + (C ↑A - 1) / E ஈ. C + D + (C - E)↑ 3 * 4

உ. A - B + C - D + F ஊ. A / B * C↑D↑(4 + 3 - 2)

4.5 நூலகச் சார்புகள்

எண்மாறிகள், எண்மாறிலிகள், எண்மானச் செயல்பாடுகள் இவற்றைப் பயன்படுத்தி எண்மானக் கோவைகளை எழுதலாம் என்பதனைப் பார்த்தோம். இவற்றுடன் பொதுவான சில **சார்புகளையும்** (Functions) பயன்படுத்தலாம். உதாரணமாக, sin .(x), log (x). x இன் தனி மதிப்பு (absolute Value of x), x இன் மூலம் (Square root) போன்ற சார்புகள் பலருக்கும் பயன்படுபவை. இவற்றைக் கணக்கிடத் தேவையான கட்டளை நிரல்கள் தொகுக்கப்பட்டு, ஒரு நூலகமாக, பேசிக் மொழிமாற்றியுடன் வரும். இவற்றை நாம் அப்படியே பயன்படுத்திக் கொள்ளலாம். இவற்றைக் கணக்கிட நாம் கட்டளை நிரல்கள் எழுத வேண்டியதில்லை. உதாரணங்கள்:-

```
LET X = R * COS (M)
LET M = LOG (A) + (ABS(K)) * EXP (3.7)
LET A = INT (Z3) + SGN(L)
```

சாதாரணமாக, பேசிக் மொழிமாற்றியுடன் வரும் முக்கியமான சில சார்புகளின் விளக்கங்கள் பின்வருமாறு.

சார்பின் பெயர்	விளக்கம்
1. SIN, COS, TAN	SIN (X) போன்றவற்றில், X ரேடியனில் கொடுக்கப்பட வேண்டும்.
2. ATN	TAN^{-1} (X) ரேடியனில் கிடைக்கும்.
3. LOG	Natural Logrithm. Base = e
4. EXP	EXP(X) = e^x.
5. SQR	SQR(X) = $\sqrt{X}$.
6. ABS	X இன் தனி மதிப்பு, ABS (5) = 5. ABS (-7) = 7.
7. INT	X அல்லது அதற்குக் குறைவாக உள்ள மிகப் பெரிய முழு எண், இது கணித இயலில் floor function எனப்படும். INT (2.3) = 2 INT (2.99) = 2 INT (-2.3) = -3 INT (-2.99) = -3
8. SGN	SGN(0) = 0 SGN *(நேர்மறை எண்) = 1.* SGN *(எதிர்மறை எண்) = -1*

9. RND *இதுபற்றிப் பின்னர் பார்ப்போம்.*

நினைவில் நிற்க

1. பொதுவாகப் பலருக்கும் பயன்படும் சில சார்புகள் நமக்கு ஆயத்தமாகக் கிடைக்கும். இவை தொகுக்கப்பட்டு ஒரு நூலகம் போல் மொழிமாற்றியுடன் வரும். இவற்றை எண்மானக் கோவைகளில் பயன்படுத்திக் கொள்ளலாம்.

2. SIN (X) போன்றவற்றில் Xஐ ரேடியனில் கொடுக்க வேண்டும்.

பயிற்சி வினாக்கள்

1. நூலகச் சார்புகள் என்றால் என்ன? சில முக்கியச் சார்புகளைப் பட்டியலிடு. அவற்றின் செயலை விளக்கு.

2. ABS, INT, SGN என்ற சார்புகள் என்ன கொடுக்கும்?

3. கீழ்வரும் கோவைகளை மதிப்பிடு.

அ. SQR (ABS (-4) + INT (12.9))

ஆ. SQR (10 + 4 + 2) + 4 * 7 / 2 + SGN (ABS (-7))

இ. -1 * INT (2 * (1 + ABS (2.3))) - 7.7

4.6. உள்ளீடு - கொடு

ஒரு கட்டளை நிரல் செயல்பட, அதற்குத் தேவையான தகவல்களை நாம் தர வேண்டும். இதை மூன்று விதங்களில் செய்யலாம். ஒவ்வொரு விதத்திலும் நிறை குறைகள் உண்டு.

முதல் முறை :- தகவல்களை கட்டளை நிரலிலேயே பதித்து வைக்கலாம். இதனால் கட்டளை நிரல் விரைவாகச் செயல்படும். ஆனால், தகவல்களை மாற்றி மறுபடியும் கணக்கிட, கட்டளை நிரலையே மாற்றி, மீண்டும் மொழி மாற்றம் செய்ய வேண்டும். இம்முறை பற்றி அடுத்த பாடத்தில் பார்ப்போம்.

இரண்டாவது முறை :- தகவல்களை ஒரு கோப்பாக மாற்றி வெளிநினைவகம் ஒன்றில் வைத்திருந்து, தேவையானபோது அதிலிருந்து எடுத்துக் கொள்வது. இது ஒரு சிறந்த முறை. தகவல்களை மாற்ற, கோப்பினை மாற்றினால் போதும். வெளி நினைவகத்தின் வேகத்தைப் பொறுத்து விரைந்து செயல்படும். இம்முறை பற்றி வேறொரு பகுதியில் காண்போம்.

மூன்றாவது முறை :- கணிப்பொறி செயல்பட்டுக் கொண்டிருக்கும்போது, தகவல் தேவையானபோது நம்மிடன் கேட்கும். நாம் சாவிப்பலகை மூலம் தகவலை அவ்வப்போது உள்ளீடு செய்யலாம். இதன் சாதக அம்சம் - தகவலை மாற்றிச் கொடுக்க இது எளிய முறை. ஆனால் ஒவ்வொரு முறையும் நம்மிடம் தகவலுக்காக கணிப்பொறி காத்துக்கொண்டிருப்பது அதன் நேரத்தை விரையமாக்கும். இம்முறையில் கணிப்பொறி எப்படித் தகவலைக் கேட்டுப் பெறும் என்பதை இப்பாடத்தில் பார்ப்போம்.

மாறிகளுக்கான மதிப்புகளைத்தான் உள்ளீடு செய்ய முடியும். ஓர் இடத்தில் A என்ற மாறிக்கான மதிப்பை உள்ளீடு செய்ய வேண்டுமானால், அந்த இடத்தில்

INPUT A

(கொடு) என்ற கட்டளையை நிரலில் எழுதவேண்டும். கணிப்பொறி இதைப் பார்க்கும்போது, உள்ளீடு தேவைப்படுகிறது என்பதைக் குறிக்க, திரையில் ஒரு கேள்விக்குறியைக் காண்பிக்கும். அப்போது நாம் சாவிப்பலகை மூலம் ஓர் எண்ணை உள்ளிட்டு, Return அல்லது Enter என்ற சாவியைத்தட்ட வேண்டும். 25 என்று உள்ளீடு செய்வதாக வைத்துக்கொள்வோம். கணிப்பொறியானது இந்த எண்ணை எடுத்து, A என்ற மாறிக்கு ஒதுக்கிய இடத்தில் 25 என்று எழுதிவைக்கும்.

ஒன்றுக்கு மேற்பட்ட தகவல்களையும் ஒரே சமயத்தில் உள்ளீடு செய்யலாம்.

10 INPUT A, B, NAME$

என்னும் கட்டளை செயல்படும்போது, கேள்வக்குறி திரையில் தோன்றியவுடன், இரண்டு எண்களையும், ஒரு சரத்தையும் உள்ளீடு செய்ய வேண்டும். உள்ளீடு செய்யும்போது கவனிக்க வேண்டிய சில விதிகள்.

1. நாம் கொடுக்கும் ஒரு தகவல் ஓர் எண்ணாக அல்லது ஒரு சரமாக இருக்க வேண்டும்.

2. INPUT கட்டளையில் கொடுத்துள்ள வரிசையிலேயே தகவல்களை நாம் உள்ளிட வேண்டும்.

3. இரண்டு தகவல்களுக்கு இடையே ஒரு காற்புள்ளி இட வேண்டும். இல்லையென்றால் இரண்டு எண்களை ஒரே எண்ணாகப் படிக்க நேரிடும்.

4. சரத்தை மேற்கோள் குறிக்குள் கொடுப்பது நல்லது. எடுத்துக்காட்டாக, 12, 23, "GOOD" என்று சாவிப்பலகை மூலம் உள்ளிடலாம். 1223 GOOD என்பது தவறு.

சாவிப்பலகை மூலம் உள்ளிடும்போது, கணிப்பொறியின் நேரம் வீணாவதால், குறைந்த அளவு உள்ளீடுசெய்வதற்கு மட்டுமே இம்முறையைப் பயன்படுத்துதல் நல்லது.

நினைவில் நிற்க

1. கட்டளை நிரல் செயல்படத் தேவையான தகவல்களை மூன்று விதங்களில் உள்ளிடலாம். அவை நிரலிலேயே வைத்தல், சாவிப்பலகை மூலம் உள்ளிடுதல், மற்றும் வெளிநினைவகங்களில் உள்ள கோப்புகளிலிருந்து எடுத்துக்கொள்ளுதல்.

2. மாறிகளின் மதிப்பினை மட்டுமே உள்ளீடு செய்யலாம்.

3. குறைந்த அளவில் உள்ளிட, சாவிப்பலகையைப் பயன்படுத்தலாம்.

4. இவ்வாறு உள்ளிடும்போது சில விதிகளைக் கடைப்பிடிக்க வேண்டும்.

பயிற்சி வினாக்கள்

1. எத்தனை விதங்களில் தகவல்கள் உள்ளீடு செய்யலாம்? அவற்றின் குறை நிறைகள் யாவை?

2. INPUT கட்டளை மூலம் உள்ளீடு செய்யும்போது கவனிக்க வேண்டிய விதிகள் யாவை?

3 கீழ்க்காணும் வாக்கியங்களில் தவறுகள் ஏதேனும் இருந்தால் கண்டுபிடி.

அ. INPUT A B C

ஆ. INPUT "A", "B", "B$"

இ. INPUT A = 23.7

ஈ. INPUT A $ B $ C $

4.7 உள்ளீடு - படி

தேவையான தகவல்களை கட்டளை நிரலிலேயே பதித்து வைக்கும் முறையை இங்கு காண்போம். இதற்கு DATA, READ, RESTORE என்னும் மூன்று வாக்கியங்களைப் பற்றித் தெரிந்துகொள்ள வேண்டும்.

DATA என்னும் வாக்கியம், இந்த வரியில் வருபவை தகவல்கள் என்பதைக் குறிக்கும். தகவல்கள் எண்களாக அல்லது சரங்களாக இருக்கலாம். ஒவ்வொன்றும் அடுத்தடுத்து வரும்போது இடையில் காற்புள்ளிகள் வர வேண்டும். உதாரணம் -

DATA 22, 54.3, MATHS, 78, PHYSICS

தகவல்கள் நிறைய இருந்து ஒரு வரி போதவில்லை என்றால் பல DATA வாக்கியங்களைப் பயன்படுத்தலாம். இவை அடுத்தடுத்து வர வேண்டும் என்ற அவசியமில்லை. இந்த வரிகள் வரி எண்களைக்கொண்டு வரிசைப்படுத்தப்பட்டு, பின்னர் தகவல்கள் எல்லாம் இந்த வரிசையில் வரிசைப்படுத்தப்படும். உதாரண்மாக, ஏற்கனவே பார்த்த DATA வாக்கியத்தை,

```
100 DATA 22, 54.3
110 DATA MATHS, 78
600 DATA PHYSICS
```

என்றும் எழுதலாம். காற்புள்ளியானது இரண்டு தகவல்களை வேறுபடுத்துக் காட்டுகிறது. சரத்திற்குள்ளே ஒரு காற்புள்ளி இருந்தால் என்ன செய்வது? MADRAS, INDIA என்பதை இரண்டு சரங்களாக அல்லவா படித்துவிடும்?

இதைத்தவிர்க்க, இதை ஒரே சரமாகப் படிக்க, இதை மேற்கோள் குறிகளுக்குள் எழுதலாம். மேற்கோள் குறிகளுக்குள் உள்ளது ஒரே சரமாகப் படிக்கப்படும்.

DATA வாக்கியங்கள் மூலம் தொகுக்கப்பட்ட தகவல்களைப் படிக்க READ data1, data2,... என்னும் வாக்கியத்தைப் பயன்படுத்த வேண்டும். data1, data2 என்பவை மாறிகள். அவற்றுக்கிடையே காற்புள்ளி வருவதைக் கவனிக்கவும். உதாரணமாக,

```
200 READ A, B, SUBJECT1$
240 READ C
260 READ SUBJECT2$
```

என்னும் கட்டளைகள், நாம் ஏற்கனவே பார்த்த தகவல் திரட்டைப் பயன் படுத்தினால், A = 22, B = 54.3, SUBJECT1$ = "MATHS", C = 78, SUBJECT2$ = "PHYSICS" என்னும் மதிப்புகளைப்பெறும். தகவல்கள் கொடுக்கப்பட்ட வரிசைப்படியே அவற்றைப் படிக்கவேண்டும். அதாவது முதல் இரண்டு, எண்கள்.

அவற்றை எண்களாக அல்லது சரங்களாகப் படிக்கலாம். ஆனால், அதற்கு அடுத்து வரும் MATHS என்பதை ஒரு சரமாகத்தான் படிக்க முடியும். அதேபோல் PHYSICS என்பதையும் ஒரு சரமாகத்தான் படிக்க முடியும். ஒரு சரத்தை ஒரு எண்ணாகப் படிக்க முயற்சி செய்தால், கணிப்பொறி இலக்கணப் பிழை (Synatx Error) என்று கூறி படிக்காமல் நின்றுவிடும். உதாரணமாக, READ A, B, C என்பது, நாம் ஏற்கனவே பார்த்த சூழ்நிலையில் தவறாகும். ஒரே READ கட்டளைமூலம் பல தகவல்களைப் படிக்க முடியும். தகவல்கள் தீர்த்து போன பிறகும், ஏதாவது ஒரு தகவலைப் படிக்கச் சொன்னால், அதுவும் தவறில் முடியும். "தகவல் தீர்ந்துவிட்டது" (Out of Data) என்ற **பிழைச்செய்தி** (Error Message) வரும். கொடுத்திருக்கும் தகவல்களையே மீண்டும் மீண்டும் படிக்கச் செய்ய ஒரு வழி இருக்கிறது. தகவல்களை ஒவ்வொன்றாகப் படிக்கும்போது, எந்தத் தகவலை அடுத்ததாகப் படிக்க வேண்டும் என்பதை ஒரு சுட்டி கணக்கு வைத்துக் கொண்டு காண்பிக்கும். கட்டளை நிரலில் எந்த இடத்திலாவது RESTORE என்று கட்டளை இட்டால், இந்தச் சுட்டியானது முதல் இடத்தைச் சுட்டிக் காட்டும்படி மாற்றப்படும். அதனால் அடுத்த தகவலைப் படிக்கும்போது திரும்பவும் முதலில் இருந்து தகவல்கள் படிக்கப்படும். RESTORE கட்டளையை எத்தனை முறைகள் வேண்டுமானாலும் பயன்படுத்தலாம்.

இந்தக் கட்டளையை RESTORE Line number என்றும் கொடுக்கலாம். உதாரணமாக, RESTORE 110 என்றால், சுட்டியானது MATHS என்ற தகவலை சுட்டிக்காண்பிக்கும்படி மாற்றப்படும். ஏனென்றால் 110 ஆம் வரியில் உள்ள முதல் தகவல் MATHS.

```
400 RESTORE 600

410 READ NAME$
```

என்றால் NAME$ என்பது PHYSICS என்ற மதிப்பைப் பெறும்.

DATA வாக்கியங்கள், READ கட்டளைகளுக்கு முன்னால் வர வேண்டுமா?- என்பது இயற்கையாக எழும் கேள்வி. இப்படி வருவது நாம் படிப்பதை எளிதாக்கும். ஆனால் அவசியமில்லை.

```
10 READ A, B, C
20 RESTORE
30 READ X, Y, Z
40 PRINT A, B, C, X, Y, Z
50 DATA 99, 88, 77
60 END
```

என்பது சரியான நிரல், இதன் வெளியீடு எவ்வாறு இருக்கும்?

பயன்படுத்தும் மொழிமாற்றியைப் பொருத்து இந்த READ, RESTORE என்னும் கட்டளையின் இலக்கணமும், பொருளும் மாறுபடும். அவற்றை நாம் பார்க்கப்போவதில்லை.

நினைவில் நிற்க

1. தகவல்களைக் கட்டளை நிரல்களுக்குள்ளேயே வைக்கலாம். இதற்கு DATA வாக்கியம் பயன்படுகிறது. இவற்றைப் படிக்க READ கட்டளை பயனாகிறது.

2. தகவல்கள் வரும் விதத்திலேயே அவற்றை வரிசையாகப் படிக்க வேண்டும்.

3. DATA மற்றும் READ வாக்கியங்கள் எத்தனை வேண்டுமானாலும் வரலாம். ஒவ்வொன்றிலும் எத்தனை தகவல்களை வேண்டுமானாலும் கொடுக்கலாம் அல்லது படிக்கலாம். இந்த வாக்கியங்கள் எங்கு வேண்டுமானாலும் வரலாம்.

பயிற்சி வினாக்கள்

1. DATA வாக்கியங்கள் மூலம் தகவல்களைத் திரட்டிவைக்கும் முறையைக் கூறு இதற்கான விதிகள் எவை?

2. READ கட்டளையின் இலக்கணத்தையும் செயல்பாட்டையும் விளக்கு.

3. RESTORE என்னும் கட்டளையின் இலக்கணத்தையும் செயல்பாட்டையும் விவரி.

4. கீழ்க்காணும் வரிகளில் தவறு ஏதேனும் இருந்தால் கண்டுபிடி.

அ. 10 READ A B

ஆ. 10 READ A$, B

இ. 20 READ A, 45

20 DATA 22, 45, "X"

ஈ. 10 RESTORE A

20 DATA A,B

30 READ X

உ. 10 READ A + B

20 DATA 20 + 40

ஊ. 10 READ A, NAME$

20 DATA 2, NAME$

எ. 10 READ A, X$

20 RESTORE

30 READ B$, C

40 DATA 5, GAME, CHESS, 7

5. கீழ்க்காணும் கட்டளைகள் செயல்படுத்தப்பட்டால் எந்தெந்த மாறி எந்தெந்த மதிப்புப் பெறும் என்று காண்.

அ. 10 READ A, MM$

20 DATA 50

30 RESTORE

40 DATA "TOY", "BALL"

50 READ X, YY$, ZZ$

ஆ. 10 READ A, B, C$

20 RESTORE 100

30 DATA 20, 30

40 READ GAME$, NAME$, WHAT$

50 RESTORE

100 DATA "CHESS", "ANAND", "CHAMPION"

110 READ M, N

இ. மேலே உள்ள வரிகளுடன்

15 DATA 40, 50, VISWANATHAN

என்னும் வரியைச் சேர்.

ஈ. மேலும் இதையும் சேர்.

25 READ WW$.

4.8 இருவழிப் பிரிதல்

இதுவரை, (1) தகவல்களை உள்ளீடு, வெளியீடு செய்யும் முறைகள், (2) மாறி, மாறிலி, நூலகச் சார்புகள், எண்மானச் செயல்பாடுகள் இவற்றைக் கொண்டு கோவைகளை உருவாக்கும் முறைகள், இந்தக் கோவைகளை மதிப்பிடும் வழிமுறை, (3) மாறிக்கு மதிப்பளிக்கும் வாக்கியம், ஆகியவற்றைப் பார்த்தோம். அதாவது தகவல்களைப் பற்றிய அடிப்படைச் செய்திகளைப் பார்த்தோம். இப்போது, கட்டளை நிரலில் எந்த வாக்கியத்திற்குப் பிறகு எந்த வாக்கியம் செயல்படுத்தப்பட வேண்டும் என்று கூறும் **கட்டுப்பாட்டு வாக்கியங்களை** (Control Statements) பற்றிப்பார்ப்போம். இவற்றில் பிரிதல், பலமுறை செய்தல் என்னும் இரண்டு வகைகளில், நிபந்தனையின்றிப் பிரிதல், இருவழிப்பிரிதல், பலவழிப்பிரிதல், நிச்சயித்துப் பலமுறை செய்தல், நிச்சயமற்று பலமுறை செய்தல் என்ற ஐந்து வித வாக்கியங்கள் உண்டு. முதல் இரண்டினை இந்தப் பாடத்தில் பார்ப்போம்.

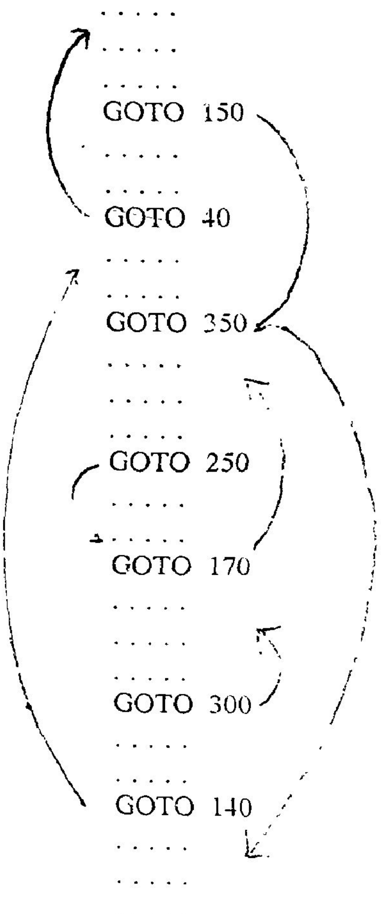
நிபந்தனையின்றிப் பிரிதல்

பிரிதலில் முதலில் நாம் தெரிந்துகொள்ள வேண்டியது GOTO (செல்) என்னும் கட்டளை, GOTO line number என்பது இதன் இலக்கணம். GOTO 580 என்றால், இந்தக் கட்டளையைப் படித்ததும், அடுத்ததாக 580 ஆம் வரி எண் கொடுக்கப்பட்ட வரிக்குச் சென்று, அங்குள்ள கட்டளையைச் செய் என்பது பொருள். சாதாரணமாக ஒரு கட்டளையைச் செய்து முடித்ததும் கணிப்பொறி யானது அடுத்த கட்டளையைப்படித்து அதன்படி செய்யும். அப்படி இல்லாமல், வரிசையாகச் செயல்படாமல், வேறு ஒரு இடத்திற்குக் கட்டுப்பாட்டைத் திசை திருப்பிவிடுவது இந்த GOTO வாக்கியத்தின் வேலை. இது **நிபந்தனையின்றிப் பிரிதல்** (Unconditional Branching) என்ற வகையைச் சேரும்.

மைக்ரோசாஃப்ட் பேசிக்கில் GOTO என்பதை நடுவில் இடைவெளி விடாமல் எழுத வேண்டும். இந்த உத்தரவை பேசிக்கில் பலமுறை பயன்படுத்துவோம். ஆனால் பாஸ்கல், ஸி போன்ற மொழிகளில் இது ஒரு தவிர்க்கப்பட வேண்டிய உத்தரவு! மிக மிகக் குறைந்த இடங்களிலேயே பயன்படுத்துவார்கள். ஏனென்றால், கட்டளை நிரலில் நிறைய GOTO உத்தரவுகள் இருந்தால் நாம் நிரலைப் புரிந்து கொள்வது கடினம்.

இருவழிப் பிரிதல்

நாம் ஏற்கனவே இருவழிப்பிரிதலை போலிக்குறியீட்டிலும், வழிப்படத்திலும் பார்த்திருக்கிறோம். அவற்றை எப்படி பேசிக் மொழியில் எழுதுவது என்று பார்ப்போம். என்றால்...என்றுசெய், இல்லையென்றால்...என்று செய் என்பதும் என்றால்..... என்று செய் என்பதும் இருவழிப் பிரிதலுக்கான போலிக் குறியீடுகள்.





"நான் இப்போ எங்கே இருக்கேன்? எங்கேருந்து வந்தேன்?"

மேம்படுத்தப்பட்ட பேசிக்கில் IF...THEN மற்றும் IF...THEN...ELSE... என்னும் வாக்கியங்கள் உண்டு. ஆனால் இவை ஒரே வரிக்குள் (சாதாரணமாக அதிகபட்சம் 255 குறியீடுகள் அல்லது 80 குறியீடுகள்) இருக்க வேண்டும். IF வாக்கியத்தைப் பற்றித் தெரிந்து கொள்வதற்கு முன்னால், அதில் பயன்படும் நிபந்தனையை பேசிக்கில் எப்படி உருவாக்குவது என்பதைப் பார்ப்போம்.

Conditional Expression என்பது **நிபந்தனை** அல்லது **நிபந்தனைக் கோவை. உறவுச் செயல்பாடுகளான** (Relational Operators) =, <> (not equal to), <, >, < = (less than or equal to), > = (greater than or equal to) போன்றவற்றை வைத்து உருவாக்கப்படுபவை நிபந்தனைகள். உதாரணங்கள்.

A < 3, (A + B) < = (C + D), C < > D.

இந்த நிபந்தனைகளை மதிப்பிட்டால் ஆம் அல்லது இல்லை என்ற இரண்டில் ஒரு மதிப்பு கிடைக்கும். இந்த உறவுச் செயல்பாடுகளை இரண்டு சரங்களுக்கு இடையிலேயும் பயன்படுத்தலாம். இங்கு < என்பது, அகராதியில் முன்னால் வரும்

என்ற பொருளில் பயனாகும். உதாரணமாக, "A" < "ABLE", "BAD" < "GOOD", "GIRL" < "MAKE" என்பவை ஆம் என்ற மதிப்பைப் பெறும். "GIRL" < "BOY", "MAN < "BOY" என்பவை இல்லை என்ற மதிப்பைப் பெறும்.

இரு நிபந்தனைகளைச் சேர்த்தும் ஒரு புது நிபந்தனையை உருவாக்கலாம். இதற்கு, தர்க்கச் செயல்பாடுகளான (Logical Operations) AND, OR என்பவற்றைப் பயன்படுத்த வேண்டும். மேலும் NOT என்னும் ஓர்உறுப்புச் செயல்பாட்டையும் பயன்படுத்தலாம். இவை நாம் ஏற்கனவே பார்த்த பூலியன் செயல்பாடுகளே. இங்கு 1 என்பது "ஆம்" (உண்மை) என்பதையும், 0 என்பது "இல்லை" (பொய்) என்பதையும் குறிக்கும். வேறு எந்த வித்தியாசமும் இல்லை.

(A > 7) மற்றும் (B > 3) என்றிருந்தால் - என்பதை

IF (A < 7) AND (B > 3) என்று எழுத வேண்டும்.

X = 77 அல்லது B$ < C$ என்பதை

IF (X = 77) OR (B$ < C$) என்று எழுதலாம்.

XX = 66 என்று இல்லாதபோது என்பதை

NOT (XX = 66) என்று எழுதலாம்.

தர்க்கச் செயல்பாடுகளிலும் முன்னுரிமை உண்டு. முதலில் NOT, அடுத்தது AND, கடைசியாக OR. அதனால், (A < 7) OR (B = 45) AND NOT (X = 6) என்பதில் NOT (X = 6) என்பது முதலிலும்,

(B = 45) AND (NOT (X = 6)) என்பது அடுத்ததாகவும், OR என்பது கடைசியிலும் செயல்படுத்தப்படும்.

(A < B) OR (B < C) AND (C < D) AND (A < R)

என்பதில் (B < C) AND (C < D) என்பது முதலிலும், பிறகு அடுத்த AND, பிறகு OR என்பவை செயல்படும்.

பேசிக் மொழியில், IF வாக்கியம் பொதுவாக,

IF conditional expression THEN statements or line number

ELSE statements or line number

IF conditional expression THEN statements or line number என்று இரு வடிவங்களில் இருக்கலாம். இங்கு statements என்று பன்மையில் இருப்பதைக் கவனிக்கவும். ஒன்றுக்கு மேற்பட்ட வாக்கியங்கள், இடையில் முக்காற் புள்ளியுடன் (:) வரலாம். எத்தனை வேண்டுமானாலும் வரலாம். ஆனால், மொத்த வாக்கியமும் ஒரு வரிக்குள் இருக்க வேண்டும். IF...THEN...ELSE வாக்கியத்தில்

இரண்டில் ஒரு பகுதி மட்டும் செயல்படுத்தப்படும். பிறகு கட்டுப்பாடு அடுத்த வரிக்குச் சென்றுவிடும். இந்த THEN பகுதி அல்லது ELSE பகுதியில் GOTO வரி எண் என்று இருந்தாலும் அல்லது வரிஎண் மட்டும் தனியாகக் கொடுக்கப்பட்டாலும், கட்டுப்பாடு அந்த வரிக்குச் சென்றுவிடும். அங்கிருந்து **தானாக** அடுத்த வரிக்குத் **திரும்பாது**. தேவையானால் நாம்தான் திசை திருப்பிவிட வேண்டும். இது சில சமயங்களில் தவிர்க்க முடியாத செயலாக இருக்கும். THEN பகுதிக்கான (அல்லது ELSE பகுதிக்கான) வாக்கியங்கள் மிக அதிகமாக இருந்தால் அவை எல்லாவற்றையும் ஒரே வரியில் எழுத முடியாது. அப்போது வேறு இடத்திற்குச் சென்று அவற்றைச் செய்வது அவசியம்.

கீழே உள்ளவை சரியான வாக்கியங்கள்.

```
IF (A < 3) AND (B > 8) THEN X = X + 1 : Y = Y + 1 ELSE X = X + 2
IF B > 7 THEN GOTO 60 ELSE M = M + 1 : Z = Z * 5 : L = L + 3
IF I < > J THEN GOTO 100 ELSE 40
IF M = N THEN 70 ELSE 700
IF B = 1 THEN X = X + 1 : Y = Y * Y ELSE 100
IF I = J THEN A = A + 1 : J + 5 : L = L + 5
IF M < = 5 THEN GOTO 70
IF L = 1 THEN 80
IF A < 7 THEN XX = YY : M = N : GOTO 100 ELSE A = 2 : GOTO 90
IF R = S THEN A = B : B = C ELSE C = D : E = F : GOTO 70
IF A = B THEN R = S : GOTO 40 ELSE 90
```

இங்கு வாக்கியங்களுக்கான அடைப்புக்குறிகள் எவையும் இல்லாவிட்டாலும், THEN முதல் ELSE வரை, அல்லது (ELSEஇல்லாவிட்டால்) THENமுதல் வரியின் இறுதிவரை, மற்றும், ELSE முதல் வரியின் இறுதிவரை உள்ள வாக்கியங்கள் அடைப்புக் குறிக்குள் உள்ளன போல் செயல்படும்.

சில வாக்கியங்களில், THENமற்றும், ELSE பகுதிகளில் சில செயல் பாடுகளைச் செய்த பிறகு கட்டுப்பாடு திசை திருப்பி விடப்படுகிறது. இவ்வாறு திசை திருப்பிவிடப்பட்டால் கட்டுப்பாடு தானாக அடுத்த வரிக்குத் திரும்பாது.

நாம் எழுதிய போலிக் குறியீடுகளில் எப்போதும், IFவாக்கியத்தில் எந்தப் பகுதி செயல்பட்டாலும் அடுத்த வாக்கியத்திற்கு வருவோம் என்று வைத்துக்கொண்டோம் .இங்குள்ள பகுதிகள் பெரியதாக இருந்தால், ஒரே வரியில் IFவாக்கியத்தை எழுத முடியாதபோது நாமாக இப்படித் திசை திருப்பிவிட வேண்டும் என்று பார்த்தோம். இதைச் செய்யும் சில உதாரணங்களைக் கீழே காண்போம். இங்கு சிறிய பகுதிகளை அப்படியே எழுதுவோம். தேவையானல் அவற்றின் முடிவில் திசை மாற்றுவோம். பெரிய பகுதிகளை வேறு இடத்திற்குச் சென்று செய்வோம். இவற்றை வேறு பல விதங்களிலும் எழுதலாம். இது ஒரு விதம்.

அ. **இரண்டு பகுதிகளும் பெரியவை.**

```
100 IF... THEN GOTO 110 ELSE 500
110 REM THEN CLAUSE

.

200 GOTO 1000
210 REM END OF THEN CLAUSE
500 REM ELSE CLAUSE


700 REM END OF ELSE CLAUSE
1000 REM NEXT STATEMENT STARTS HERE

.
```

ஆ. THEN **பகுதி மட்டும் பெரியது.**

```
100 IF... THEN GOTO 110 ELSE A=B: GOTO 210
110 REM THEN CLAUSE

.
200 REM END OF THEN CLAUSE
210 REM NEXT STATEMENT STARTS HERE.
```

இ. ELSE **பகுதி மட்டும் பெரியது.**

```
100 IF .. THEN A=B: GOTO 210 ELSE GOTO 110
110 REM ELSE CLAUSE

.
200 REM END OF ELSE CLAUSE
210 REM NEXT STATEMENT STARTS HERE.
```

ஈ. ELSE **பகுதியே இல்லை,** THEN **பகுதி பெரியது.**

```
100 IF... THEN GOTO 110 ELSE GOTO 500

110 REM THEN CLAUSE
```

```
400 REM END OF THEN CLAUSE
500 REM NEXT STATEMENT STARTS HERE.
```

GOTO வாக்கியத்தைப் பயன்படுத்தும்போது அங்கிருந்து போக வேண்டிய வரியின் வரி எண் உடனே தெரியாமலிருக்கலாம். சில வரிகள் எழுதிய பிறகே அந்த வரி எண் தெரிய வரும். அதனால், முதலில் இந்த வரி எண்களை எழுதாமல் விட்டுவிட்டு, பிறகு அந்த இடங்களை (மறக்காமல்) பூர்த்தி செய்வது நல்லது.

THEN மற்றும் ELSE பகுதிகளைத் தனியாக வேறுவரிகளில் எழுதும்போது இடதுபக்கம் சற்று இடம் விட்டு எழுதுவது நாம் புரிந்துகொள்வதை எளிதாக்கும்.

மேலே நான்கு விதங்களில் திசை திருப்பி விடப்பட்டிருந்தாலும், ஒருவர் விரும்பினால், முதல் முறை ஒன்றை மட்டுமே எப்போதும் பயன்படுத்தலாம். அது ஒன்றே போதுமானது, எளிதானதும் கூட.

சிறு கட்டளை நிரல்கள் எழுதத் தயாராகிவிட்டோம். இங்கு இரண்டு நிரல்கள் எழுதுவோம். ஒவ்வொரு பாடத்திலும், புதிதாகக் கற்றுக் கொண்டதைத் தெளிவாக்கிக் கொள்ள இப்படி ஒன்றிரண்டு நிரல்கள் எழுதுவோம். பயிற்சி வினாக்களில் மேலும் பல கணக்குகளைப் பார்க்கலாம்.

1. வீட்டு உபயோக மின் கட்டணம், முதல் நூறு அலகுகள் (Units) வரை அலகுக்கு 60 காசுகள். அதற்கு மேல் ஒவ்வொரு அலகுக்கும் 1 ரூபாய். எத்தனை அலகுகள் பயன்படுத்தப்பட்டன எனக் கேட்டு வாங்கி, அதற்கான கட்டணத்தைக் கண்டுபிடி.

```
10 PRINT "GIVE NUMBER OF UNITS"
20 INPUT UNITS
30 IF UNITS < = 100 THEN A = UNITS * 0.6 : GOTO 70 ELSE 40
40 REM ELSE CLAUSE
50 A = (100 * 0.6) + UNITS - 100
60 REM END OF ELSE
70 PRINT "TOTAL AMOUNT = ", A
```

இந்த நிரலில் IF வாக்கியத்தை ஒரே வரியில் எழுதியிருக்கலாம். அது கீழ்க்காணுமாறு இருக்கும்.

```
IF UNITS < = 100 THEN A = UNITS * 0.6 ELSE A = (100 * 0.6) + UNITS - 100
```

கட்டளை நிரலில் கடைசி வாக்கியமாக END என்று எழுத வேண்டும். இது கட்டளை நிரல் முடிவடைந்ததைக் காட்டும். அதனால் செயல்பாடு முடிவடைந்ததையும் குறிக்கும். செயல்பாடு முடிவடைந்ததை மட்டும் குறிப்பிட STOP என்ற வாக்கியமும் பயன்படும். ஒரு நிரலில் STOP என்ற வாக்கியம் பல இடங்களில் வரலாம். ஆனால், சாதாரணமாக END என்பது ஒரே ஒரு முறை

தான் வரவேண்டும். சில மொழிமாற்றிகளில் END பல முறை வரலாம். STOP என்பது, நாம் **முற்றும்** என்று போலிக்குறியீடுகளில் பல இடங்களில் எழுதிய தற்குச் சமமான பேசிக் கட்டளை.

2. "உன் வயது என்ன?" என்று கேட்டு, ஒரு எண்ணை வாங்கு. அந்த எண் 0 அல்லது எதிர்மறை என்றால் "நாம் கனவுலகத்திலா இருக்கிறோம்?" என்று எழுது. எண் பத்துக்குக் கீழே இருக்கிறதென்றால் "நீ சின்னக் குழந்தை" என்று எழுது, இல்லையென்றால் "நல்வரவு" என எழுது.

```
10 PRINT "WHAT IS YOUR AGE?"

20 INPUT AGE

30 IF AGE < = 0 PRINT "ARE WE IN A DREAMLAND?" : GOTO 100

40 IF AGE < 10 PRINT "YOU ARE A KID" : GOTO 100

50 IF AGE > = 10 PRINT "WELCOME"

100 END
```

இங்கு இரண்டு இடங்களில் GOTO 100 என்று இருக்கிறது. அவை இல்லா விட்டாலும் இந்த நிரல் சரியாக வேலைசெய்யும். ஏன்? 50வது வரியில் PRINT "WELCOME" என்று இருந்தாலும் இந்த நிரல் சரியாக வேலை செய்யும். ஏன்? ஆனால் இதுவே இரண்டு GOTOக்களை எடுத்த பிறகு சரியாக வேலை செய்யாது. ஏன்? பலவித எண்களுக்கு உலா வந்து பார்.

ஓர் எச்சரிக்கை

உறவுச் செயல்பாடான சமம் (=) என்பதைப் பயன்படுத்தும்போது சற்று எச்சரிக்கையுடன் செயல்பட வேண்டும். IF (A = B) என்பதில் இரண்டும் முழு எண்களாக இருந்தால் பரவாயில்லை. அவை பின்னங்களாக இருந்தால், மிதக்கும் புள்ளி முறையில் உள்ளே வைக்கப்பட்டிருக்கும். பின்னங்களைப் பயன்படுத்தி பல கணிதச் செயல்பாடுகளைச் செய்து இந்த மதிப்புகளைக் கணக்கிட்டிருந்தால், வரும் விடைகள் உண்மையான மதிப்பை இழந்து சற்றேறக்குறைய உள்ள மதிப்புடன் இருக்கும் என பார்த்திருக்கிறோம். அதனால், A யும் Bயும் கணக்கியலின்படி ஒரே மதிப்புப் பெற்றிருந்தாலும், கணிப்பொறிக்குள் மிதக்கும் புள்ளி முறையில் சமமில்லாத எண்களாக வைக்கப்பட்டிருக்கலாம். அதனால் இரு பின்னங்களை சமமா என்று பார்ப்பது சரியான செயலில்லை. இரண்டு பின்னங்களின் வித்தியாசம் மிகமிகக் குறைவாக இருந்தால் அவற்றை சமம் என்று எடுத்துக்கொள்ளலாம். இதை IF (ABS (A - B) < 0.000001) என்று எழுதலாம்.

நினைவில் நிற்க

1. உறவுச் செயல்பாடுகளையும், தர்க்கச் செயல்பாடுகளையும் பயன்படுத்தி நிபந்தனைக் கோவைகளை எழுதலாம்.

2. நிபந்தனைக் கோவைகளை மதிப்பிட்டால் ஆம் (உண்மை) அல்லது இல்லை (பொய்) என்ற மதிப்பு கிடைக்கும்.

3. IF...THEN மற்றும் IF...THEN...ELSE வாக்கியங்களை ஒரே வரியில் எழுத வேண்டும்.

4. பின்னங்களை சமமா என்று சரிபார்த்தல் கூடாது.

பயிற்சி வினாக்கள்

1. நிபந்தனை என்றால் என்ன? அதை மதிப்பிட்டால் என்ன கிடைக்கும்?
2. உறவுச் செயல்பாடுகள் எவை? உதாரணங்கள் கொடு.
3. தர்க்கச் செயல்பாடுகள் எவை? அவற்றுக்குள் உள்ள முன்னுரிமை என்ன?
4. IF...THEN...ELSE வாக்கியத்தை விவரி. உதாரணம் கொடு. அதன் செயல்பாட்டை ஒரு வழிப்படத்தின் மூலம் விளக்கு.
5. IF...THEN வாக்கியத்தை எழுதும் முறையையும், அதன் செயல்பாட்டையும் விவரி.
6. போலிக் குறியீட்டிலிருந்து, இருவழிப் பிரிதலை எவ்வாறு பேசிக் மொழியில் மாற்ற வேண்டும் என்பதை உதாரணத்துடன் விளக்கு.
7. கீழே உள்ள கணிப்புகளுக்குக் கட்டளை நிரல்கள் எழுது.

   அ. ஓர் எண்ணைப் படி. அதை ஃபாரன்ஹீட்டிலிருந்து செல்ஷியசுக்கு, அல்லது, செல்ஷியசிலிருந்து ஃபாரன்ஹீட்டிற்கு, எந்த விதத்தில் மாற்ற வேண்டும் என்று கேட்டு வாங்கு. C என்று உள்ளீடு செய்தால் செல்ஷியசாகவும், F என்றால் ஃபாரன்ஹீட்டாகவும் மாற்றி வெளியீடு செய்.

   ஆ. X, Y என்ற எண்களைக் கேட்டுப்பெறு. (X, Y) என்னும் புள்ளி $X^2 + Y^2 = 4$ என்ற வட்டத்திற்கு உள்ளே, வெளியே அல்லது வட்டத்திற்கு மேலே உள்ளதா என்று கண்டுபிடித்துக் கூறு.

   இ. மூன்று பக்கங்களின் நீளங்களைப் படி. அவை ஒரு முக்கோணத்தை உருவாக்குமா என்று பார். (எண்கள் சுழியாகவும், எதிர்மறையாகவும் இருக்கக் கூடாது. ஏதாவது இரண்டின் கூட்டுத் தொகை மூன்றாவதை விடக் கூடுதலாக இருக்க வேண்டும்.)

8. STOP என்பதற்கும் END என்பதற்கும் உள்ள இரண்டு வித்தியாசங்களைக் கூறு.

4.9 பலவழிப் பிரிதல்

இருவழிப் பிரிதலுக்கு ஒரு நிபந்தனைக் கோவையைப் பயன்படுத்துகிறோம். உண்மை, பொய் என்பதை வைத்து இரண்டு கிளைப்பாதைகளில் ஒன்றில் செல்கிறோம். பலவழிப் பிரிதலுக்கு இந்த முறை ஒத்துவராது. இங்கு ஓர் எண் மானக் கோவையின் மதிப்பைக் கண்டுபிடித்து, அது 1 என்றால் ஓர் இடத்திற்கும், 2 என்றால் இன்னொரு இடத்திற்கும், 3 என்றால் வேறொரு இடத்திற்கும் என்று பல இடங்களுக்குச் செல்ல வேண்டும்.

உதாரணமாக, I என்ற எண்
1 என்றால் 100 க்குப்போ,
2 என்றால் 200க்குப்போ,
3 என்றால் 500க்குப்போ,
4 என்றால் 1000க்குப்போ,

என்பதை, பேசிக் மொழியில்

```
ON I GOTO 100, 200, 500, 1000
```

என எழுத வேண்டும்.

இங்கு I என்பது ஓர் எண். இந்த எண்ணைப் பெற ஒரு மாறியை அல்லது எண்மானக் கோவையைப் பயன்படுத்தலாம். இது முழு எண்ணாக இருக்க வேண்டும் என்பது அவசியமில்லை. முழு எண்ணாக இல்லாவிட்டால், அதை அருகிலுள்ள முழு எண்ணாக மாற்றி எடுத்துக் கொள்ளும். உதாரணமாக 2.6 என்பதை 3 என்று எடுத்துக்கொள்ளும்.

கிடைக்கும் மதிப்பு 0 என்றால், இந்த வாக்கியத்தில் கொடுத்துள்ள எந்த வரிக்கும் போகாமல் அடுத்த வரிக்குப் போய்விடும். கிடைக்கும் மதிப்பு எதிர்மறை என்றால் ஒரு பிழைச் செய்தியைக் கொடுத்து விட்டு செயல்பாடு நின்றுவிடும்.

இந்த வாக்கியத்தில் உள்ள வரி எண்கள் எல்லாம் வெவ்வெறாக இருக்க வேண்டும் என்ற அவசியம் இல்லை. சில எண்கள் சமமாக இருக்கலாம்.

உதாரணம்.

```
ON K + 2 GOTO 40, 80, 60, 80, 100
```

இந்த வரி எண்கள் ஏறு வரிசையில் இருக்க வேண்டிய அவசியம் இல்லை.

ON GOTO என்ற வாக்கியத்தைப் பயன்படுத்தாமல் IF தொடர் மூலமே தேவையானபடி பிரிந்து செல்லலாம். வாக்கியங்கள் கூடுதலாக வரும். அவ்வளவுதான். சற்றுமுன் பார்த்த வாக்கியத்தை,

```
IF ( INT ( K + 2.5 ) = 1 ) THEN 40
IF ( INT ( K + 2.5 ) = 2 ) OR ( INT( K + 2.5 ) = 3 ) THEN 80
IF ( INT ( K + 2.5 ) = 4 ) THEN 60
IF ( INT ( K + 2.5 ) = 5 ) THEN 100
```

என்று எழுதலாம். இங்கு K என்பது ஒரு எண் மாறி என வைத்துக் கொண்டுள்ளோம். சில கட்டளை நிரல்களை எழுதுவோம்.

ஒரு எண்ணைப்படி. அது 1 என்றால் ONE என்று ஒரு முறை எழுது. 2 என்றால் TWO என்று இரண்டு முறைகள் எழுது. 3 என்றால் THREE என்று மூன்று முறைகள் எழுது.

```
10 PRINT "GIVE A NUMBER"
20 INPUT NUM
30 ON NUM GOTO 100, 200, 300
40 STOP
100 PRINT "ONE"
110 STOP
200 PRINT "TWO"
210 PRINT "TWO"
220 STOP
300 PRINT "THREE"
310 PRINT "THREE"
320 PRINT "THREE"
330 END
```

முதல் மற்றும் வருடங்களைப் படி. வட்டிவிகிதக் குறியீட்டைப் படி. இது 1 என்றால் வட்டிவிகிதம் 5%; 2 என்றால் வட்டிவிகிதம் 15%; 3 என்றால் 20%; 4 என்றால் 30%; தவறான வட்டிக்குறியீடு என்றால் திரும்பவும் வட்டிக்குறியீட்டைக் கேட்டுப்பெறு. தனி வட்டியைக் கணக்கிட்டு, கொடுத்த வருட முடிவில் எவ்வளவு தொகை திரும்பச் செலுத்த வேண்டும் என்று கணக்கிட்டுக்கூறு.

```
10 PRINT "GIVE PRINCIPAL AND YEARS"
20 INPUT P, YEARS
30 PRINT "GIVE INTEREST CODE"
40 INPUT ICODE
```

```
  50 ON ICODE GOTO 100, 200, 300, 400
  55 REM ERROR IN INTEREST CODE
  60 GOTO 30
 100 TOTAL = P + P * YAERS * 0.05
 110 GOTO 1000
 200 TOTAL = P + P * YEARS * 0.15
 210 GOTO 1000
 300 TOTAL = P + P * YEARS * 0.2
 310 GOTO 1000
 400 TOTAL = P + P * YEARS * 0.3
1000 PRINT TOTAL
1010 END
```

நினைவில் நிற்க

1. பலவழிப் பிரிதலுக்கு ON...GOTO கட்டளையைப் பயன்படுத்த வேண்டும். இதில் ஓர் எண்கோவை பயன்படுத்தப்படுகிறது.

2. இதில் வரும் வரி எண்கள் சமமாக இருக்கலாம். அவை ஏறு வரிசையில் இருக்கவேண்டியதில்லை.

பயிற்சி வினாக்கள்

1. a,b,c என்ற எண்களைப் படி. b^2 - 4ac என்பது எதிர்மறை எண் என்றால் 'இமாஜினரி விடைகள்' என்று எழுது; பூஜ்யம் என்றால், 'இரண்டு சமமான விடைகள்', என்றும், நேர்மறை என்றால், 'இரண்டு வேறான விடைகள்', என்றும் எழுது.

   (ON.... GOTO வாக்கியத்தில், 2 + SGN (b^2 - 4ac) என்பதைப் பயன்படுத்து).

2. ஒரு மதிப்பெண் 40க்குக் கீழே என்றால், தேர்வில் தோல்வி என்றும்,

   40 முதல் 49 இருந்தால் மூன்றாம் வகுப்பு,

   50 முதல் 59 வரை இருந்தால் இரண்டாம் வகுப்பு,

   60 அல்லது அதற்கு மேலும் இருந்தால் முதல் வகுப்பு என்றும் பிரிக்க வேண்டும். இதைச் செய்ய ON.... GOTO பயன்படுத்தாமல் ஒரு கட்டளை நிரலையும், பயன்படுத்தி ஒரு கட்டளை நிரலையும் எழுது.

3. ON.... GOTO என்ற செயல்பாட்டினை ஓர் உதாரணத்துடன் விளக்கு.

4.10 நிச்சயமற்றுப் பலமுறை செய்தல்

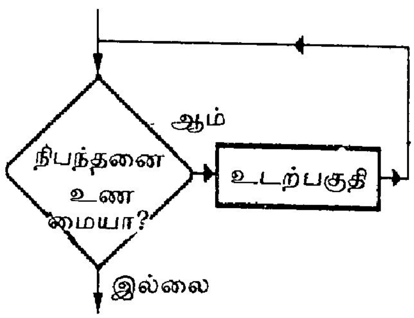
நிச்சயமற்றுப் பலமுறை செய்வதற்கான கட்டளை, மேம்படுத்தப்பட்ட பேசிக்கில் உண்டு. மற்ற பேசிக் மொழிமாற்றிகளில் இதை IF தொடரை வைத்தே செயல்படுத்த வேண்டியிருக்கும்.

WHILE (Conditional Expression *நிபந்தனைக் கோவை)*

(Body of while *உடற்பகுதி)*

WEND

என்று எழுத வேண்டும். இதன் வழிப்படம் படம் *4.1* இல் உள்ளது. WEND என்பது END OF WHILE என்பதன் சுருக்கம். இங்கு WHILE உம் WEND உம் வாக்கியங்களுக்கான அடைப்புக் குறிகள் போல் செயல்படுகின்றன.



படம் 4 - 1 நிச்சயமற்று பலமறை செய்தல்

ஓர் உதாரணத்தின் மூலம் இவ்வாக்கியத்தைப் புரிந்துகொள்வோம்.

ஒரு கோளத்தின் ஆரத்தைப் படித்து, அதன் கொள்ளளவைக் கண்டுபிடி. கொடுக்கப்படும் எண் எதிர்மறையாகவோ அல்லது சுழியாகவோ இருந்தால், கொடுத்த எண் தவறு என்று கூறி மீண்டும் ஆரத்தைக் கேள். நேர்மறை எண் கொடுக்கப்படும் வரை இப்படிச் செய்.

```
10 PRINT "GIVE A POSITIVE NUMBER"
20 INPUT RADIUS
30 WHILE (RADIUS < = 0)
40        PRINT "THIS IS NOT CORRECT"
50        PRINT "GIVE A POSITIVE NUMBER"
60 WEND
```

```
70 REM
80 VOL = (4/3) *3.14* RADIUS ↑ 3
90 PRINT "VOLUME = ", VOL
100 END
```

ஒரு பந்து, ஒவ்வொரு முறையும் போடப்பட்ட உயரத்தில் பாதிக்கு எழும்பும். போடப்படும் உயரம் 2 செ.மீக்கு அதிகமாக இருந்தால் இப்படி எழும்பும். இல்லையென்றால் எழும்பாமல் நின்றுவிடும். முதலில், பந்து போடப் படும் உயரத்தைக் கேட்டு வாங்கு. அந்தப் பந்து எத்தனை முறை எழும்பும் என்பதையும், ஒவ்வொரு முறையும் எவ்வளவு உயரம் எழும் என்பதையும் பட்டியலிடு.

```
10 PRINT "GIVE INITIAL HEIGHT IN CENTIMETERS"
20 INPUT HEIGHT
30 COUNT = 0
40 WHILE (HEIGHT > 2)
50          HEIGHT = HEIGHT / 2
60          COUNT = COUNT + 1
70          PRINT COUNT, HEIGHT
80 WEND
90 END
```

WHILE மற்றும் WEND வரிகளுக்கு இடையில் உள்ள உடற்பகுதி சற்று இடம் விட்டு எழுதப்பட்டிருப்பதைக் கவனித்தீர்களா? இது நாம் நிரலை படித்துப் புரிந்துகொள்வதை எளிதாக்கும்.

கீழே உள்ள கட்டளைகளில் ஒரு தவறு இருக்கிறது. அது என்ன?

```
100 WHILE (A < 10) AND (X < 10)
110          B = B + C
120          C = C + 2
130 WEND
```

நூறாவது வரியில் A, X இன் மதிப்புகள் பத்துக்குக் கீழே இருந்தால், இதில் உள்ள உடற்பகுதி முடிவில்லாமல் திரும்பத்திரும்பச் செய்யப்படும். ஏன்? நிபந்தனையில் உள்ள எந்த மாறியும் (A, X) உடற்பகுதியில் மாறவேயில்லை. அதனால், ஒரு முறை நிபந்தனைக்கு "உண்மை" என்று விடை கிடைத்தால், அது மாறாமல், ஒவ்வொரு முறையும் "உண்மை" என்று விடை கிடைக்கும். அதனால்,

நிபந்தனையில் வரும் ஒரு மாறியின் மதிப்பையாவது, உடற்பகுதியில் மாற்ற வேண்டும்.

ஒரு WHILE வாக்கியத்திற்குள் இன்னொரு WHILE வாக்கியம் இருக்கலாம். அப்போது ஒன்றுக்குள் மற்றொன்று முழுவதுமாக இருக்க வேண்டும். உதாரணமாக,

ஓர் எண்ணைப் படி. அது 1 முதல் 9க்குள் இருந்தால், Y அல்லது N என்ற மதிப்பில் ஒன்றைக் கேட்டுப்பெறு. மற்ற குறியீடுகளை நிராகரித்துவிடு. எண்ணையும், குறியீட்டையும் வெளியிடு. திரும்பவும் ஓர் எண்ணைப் படி. இந்த எண் 1 முதல் 9 வரை இருக்கும் வரை மேலே சொன்னதைத் திரும்பத் திரும்பச் செய். இல்லையென்றால் செயல்பாட்டை நிறுத்திவிடு.

```
10 PRINT "GIVE A NUMBER"
20 INPUT A
30 WHILE (A > 0) AND (A < 10)
40        PRINT "GIVE Y OR N"
50        INPUT CHOICES
60        WHILE NOT ((CHOICES  =  "Y") OR (CHOICE$  = "N"))
70               PRINT "GIVE Y OR N"
80               INPUT CHOICE$
90        WEND
100       PRINT A, CHOICE$
110       PRINT "GIVE A NUMBER"
120       INPUT A
130 WEND
140 END
```

நினைவில் நிற்க

1. நிச்சயமற்றுப் பலமுறை செய்வதற்கான கட்டளை WHILE.

2. இதில் நிபந்தனையில் உள்ள மாறிகளில் ஒன்றையாவது உடற்பகுதியில் மாற்றி புதிய மதிப்புக் கொடுக்க வேண்டும்.

3. ஒரு WHILEக்குள் இன்னொரு WHILE இருக்கலாம். முழுவதுமாக உள்ளே இருக்க வேண்டும்.

பயிற்சி வினாக்கள்

1. WHILE வாக்கியத்தின் செயல்பாட்டை ஒரு வழிப்படத்தின் மூலம் விளக்கு.

2. WHILE வாக்கியத்தில் முக்கியமாகக் கவனிக்க வேண்டிய விதிகள் இரண்டினைக் கூறு.

3. கீழே உள்ள கட்டளைத் தொடர்களில் என்ன தவறு?

அ.
```
10 INPUT A, B
20 WHILE A > 10
30        B = B + 2
40        A = A + 2
50 WEND
```

ஆ.
```
100 WHILE (X = 7)
110       MAX = MAX + 2
120       I = I/2
130       M$ = "NEW"
140 WEND
```

இ.
```
10 X = 1
20 WHILE (2 < X)
30        M = 43
40 WEND
```

ஈ.
```
10 X = 1 : Y = 2
20 WHILE X < 5
30 WHILE Y < 6
40        X = X + 0.5 :  Y = Y + 0.35
50 WEND
```

4. எந்த எண்ணின் மூன்றாம் அடுக்கு 8964ஐ முதலில் தாண்டிச் செல்கிறது என்பதைக் கண்டுபிடிக்க WHILE வாக்கியத்தைக் கொண்டு ஒரு கட்டளை நிரலும், அது இல்லாமல் ஒரு கட்டளை நிரலும் எழுது.

4.11 நிச்சயித்துப் பலமுறை செய்தல்

நிச்சயித்துப் பலமுறை செய்வதற்கு, பேசிக்கில் FOR என்ற வாக்கியம் உள்ளது. இது FOR **வளையம்** (FOR Loop) எனப்படும். இது அடிக்கடி பயன்படுத்தப்படும் மிக முக்கியமான வாக்கியம். இங்கு எண்ணுவதற்காக ஒரு பெயரைப் பயன்படுத்த வேண்டும் என்பதைச் சென்ற பகுதியில் பார்த்திருக்கிறோம். மேலும் ஆரம்ப மதிப்பு, இறுதி மதிப்பையும் கொடுக்க வேண்டும். சாதாரணமாக எண்ணும் பெயரின் மதிப்பில் ஒவ்வொரு முறையும் ஒன்றைக் கூட்டுவோம். FOR தொடரில் இப்படி ஒன்றைக் கூட்டலாம், அல்லது நாம் கொடுக்கும் எண்ணைக் கூட்டலாம். இந்த FOR தொடரின் பொது வடிவம்.

```
FOR index = initial value TO final value STEP stepsize
*
*
*
NEXT Index
```

இங்கு ஆரம்ப மதிப்பும், இறுதி மதிப்பும் எண்மானக் கோவைகளாக இருக்க வேண்டும். Stepsize என்பது ஒவ்வொரு முறையும் எண்ணும் மாறியுடன் (index) எவ்வளவு கூட்ட வேண்டும் என்பதைக் குறிக்கிறது. FOR வளையத்தின் உடற்பகுதி எனப்படும். இங்கு FOR, NEXT என்னும் வரிகள் வாக்கியங்களுக்கான அடைப்புக் குறிகளாக செயல்படுகின்றன. உடற்பகுதியைச் சற்று இடம் விட்டு எழுதுவது நாம் படிப்பதை எளிதாக்கும். NEXT என்பது FOR என்பதற்கு நேர்கீழாக வரவேண்டும். உதாரணம் -

```
FOR I = 1 TO 5
      M = M + I : N = N + 2*I
NEXT I
```

என்பது சரியான FOR வளையம்.

```
FOR I = 1.3 TO 7.9 STEP 0.1
FOR J = 7.2 TO -3.3 STEP -1.3 + X
FOR M = -20 TO 40 STEP 4
FOR L = A  +  B TO B 2 STEP -2.
```

என்பவை எல்லாமே சரியான வாக்கியங்கள்.

FOR வளையத்திற்கான விதிகள் சில பின்வருமாறு.

1. எண்ணும் மாறி ஓர் எண்மாறியாகத்தான் இருக்க வேண்டும்; வேறு எதுவும் கூடாது.

2. முதல் மதிப்பு, இறுதி மதிப்பு, எண்ணும் மாறியுடன் கூட்டும் தொகை இவை எல்லாம் எண்மானக் கோவைகளாக இருக்க வேண்டும். இவை ஒவ்வொன் றும், ஆரம்பத்தில் ஒருமுறை மட்டும் மதிப்பிடப்படும். அந்த மதிப்புகளையே தேவையானபோதெல்லாம் பயன்படுத்திக் கொள்ளும்.

3. இவை எல்லாமே சுழி அல்லது எதிர்மறையாகக் கூட இருக்கலாம்.

4. மாறியுடன் கூட்டும் மதிப்பு சுழி என்று இருக்கலாம். ஆனால் இதைத் தவிர்ப்பது நல்லது.

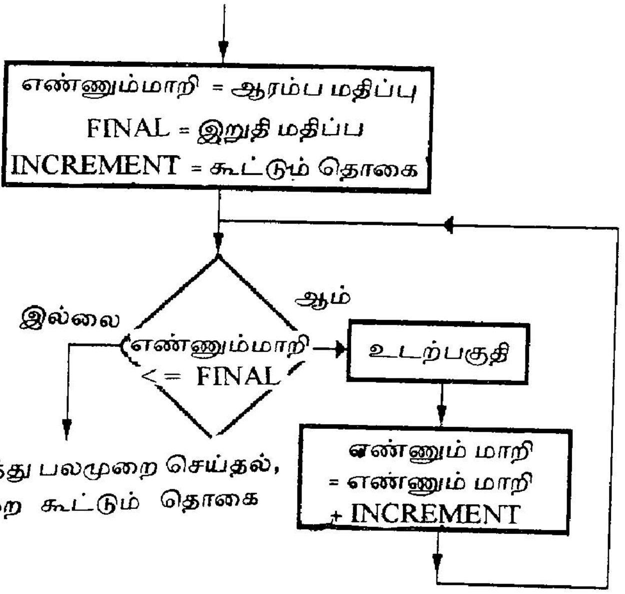
5. இந்த கூட்டும் மதிப்பு கொடுக்கப்பட வேண்டிய அவசியம் இல்லை. கொடுக்கப்படாவிட்டால், தானாக 1 என்ற மதிப்பைப் பெறும்.

6. முதல் மதிப்பு இறுதி மதிப்பை விட அதிகமாகவும், கூட்டும் மதிப்பு சுழி அல்லது நேர்மறையாகவும் இருந்தால், உடற்பகுதி ஒருமுறை கூடச் செயல்படாது.

7. முதல் மதிப்பும் இறுதி மதிப்பும் சமமாக இருந்தால், உடற்பகுதி ஒருமுறை செயல்படுமா என்பது கூட்டும் மதிப்பைப் பொறுத்தது. கூட்டும் மதிப்பு சுழி யில்லை என்றால் ஒருமுறை செயல்படும்; சுழி என்றால் ஒரு முறை கூடச் செயல்படாது.

8. முதல் மதிப்பு இறுதி மதிப்பைவிடக் குறைவாகவும், கூட்டும் மதிப்பு சுழி அல்லது எதிர்மறையாகவும் இருந்தால், உடற்பகுதி ஒருமுறை கூடச் செயல்படாது.

9. FOR வளையம் செயல்படும் விதத்தைக் குறிக்கும் வழிப்படத்தைப் படம் 4.2 இல் பார்க்கலாம். இங்கு, கூட்டும் தொகை ஒரு நேர்மறை எண் என எடுத்துக் கொள்ளப்பட்டுள்ளது. அதனால் எண்ணும் மாறி <= FINAL என்னும் நிபந்தனை சரிபார்க்கப்படுகிறது. கூட்டும் தொகை எதிர்மறை எண்ணாக இருந்தால் எண்ணும் மாறி > = FINAL என்பது சரிபார்க்கப்படும். ஏனென்றால் இங்கு எண்ணும் மாறியின் மதிப்பு குறைந்து கொண்டே வரும்.



படம் 4 - 2 நிச்சயித்து பலமுறை செய்தல், நேர்மறை கூட்டும் தொகை

10. எத்தனை முறைகள் உடற்பகுதி செயல்படுத்தப்பட்டது என்பதைக் கணக்கு வைத்துக்கொள்ள எண்ணும் மாறி பயன்படுகிறது. அதனால் அதன் மதிப்பை வளையத்தின் உடற்பகுதியில் மாற்றக்கூடாது.

```
10 FOR I = 1 TO 10
20     A = A + 2
30     I = I - 1
40 NEXT I
```

என்பது தவறு.

11. FOR வளையத்திற்கு உள்ளேயிருந்து GOTO, IF, ON... GOTO உத்தரவுகள் மூலம் கட்டுப்பாட்டை வெளியே கொண்டு வரலாம். ஆனால், இவை மூலம் கட்டுப்பாட்டை வளையத்தின் வெளியிலிருந்து உடற்பகுதிக்கு மாற்றக்கூடாது. இந்த வளையத்திற்குள் FOR வரியின் மூலம் மட்டும்தான் நுழைய வேண்டும் அப்போதுதான் சரியாகக் கணக்கு வைத்துக்கொள்ள முடியும்.

```
10 FOR I = 1 TO 10
20          IF (A - I + 3 > 20) THEN 800
30 NEXT I
```

என்பது சரி. ஆனால்

```
10 FOR I = 10 TO 20
20          K = K + I
30 NEXT I
100 GOTO 20
```

என்பது தவறு.

12. ஒரு FOR வளையத்திற்கு உள்ளே இன்னொரு FOR வளையம் முழுவதுமாக உள்ளே இருக்கலாம். ஆனால் ஒன்றை ஒன்று வெட்டிக்கொண்டு, பாதி உள்ளேயும் பாதி வெளியேயும் இருக்கக் கூடாது.

```
100 FOR I = 10 TO 20
110          A = A + B
120          FOR J = 2 TO 5
130                B = B + 2 + J
140          NEXT J
150 NEXT I
```

என்பது சரி. இங்கு 110 ஆம் வரி 11 முறை செயல்படும். 130 ஆம் எண் வரி 11 x 4 = 44 முறைகள் செயல்படும்; முதலில் I என்பது 10 என்றும், J என்பது 2, 3, 4, 5 என்றும் நான்கு முறைகள் செயல்படும். இதேபோல் I = 11, 12,.. 20 என்று

ஒவ்வொரு முறைக்கும், J நான்கு மதிப்புகளைப் பெறும். ஆகவே மொத்தம் 11 x 4 முறைகள் செயல்படும்.

```
100 FOR I = 1 TO 10
110        FOX = FOX + 1
120        FOR J = 5 TO 7
               M = M+J
130 NEXT I
140        NEXT J
```

என்பது தவறு.

13. FOR வளையத்திற்குள் எண்ணும் மாறியின் மதிப்பை மாற்றக்கூடாது என்று பார்த்தோம். ஆனால் அதன் மதிப்பை நன்கு பயன்படுத்திக் கொள்ள வேண்டும். அப்படிப் பயன்படுத்தவில்லை என்றால், எழுதிய நிரல் சரிதானா என்று ஒருமுறை சரிபார்ப்பது நல்லது.

சில உதாரணங்களைப் பார்ப்போம்.

கொடுக்கும் எண்ணிற்கான பெருக்கல் வாய்ப்பாட்டை அச்சிடு.

```
10 PRINT "GIVE AN INTEGER"
20 INPUT N
30 FOR I = 1 TO 16
40    PRINT N, "X", I, "=", N*I
50 NEXT I
60 END
```

கொடுக்கும் இரண்டு எண்களுக்கு இடையே உள்ள அத்தனைப் பெருக்கல் வாய்ப்பாடுகளையும் அச்சிடு. உதாரணமாக, 2,5 (அல்லது 5,2) என்று கொடுத்தால், 2,3,4,5 ஆம் வாய்ப்பாடுகளை அச்சிட வேண்டும்.

```
10 PRINT  "GIVE TWO INTEGERS"
20 INPUT A, B
30 IF A < B THEN 100
40 REM HERE A > = B  HENCE INTERCHANGE A  AND  B
50 TEMP  = A
60 A = B
70 B  = TEMP
80 REM
```

```
100 FOR I = A TO B
110         FOR J = 1 TO 16
120             PRINT I, "X",  J,  " = " ,  I*J
130         NEXT J
140         PRINT "——"
150 NEXT I
160 END
```

ஒரு கார் முதல் மணியில் 40 கி.மீ. வேகத்திலும், அடுத்த மணியில் 50 கி.மீ. வேகத்திலும் செல்கிறது. ஒவ்வொரு கால் மணிநேர முடிவிலும் அந்தக் கார் எவ்வளவு தொலைவில் இருக்கும் என்பதைப் பட்டியலிட ஒரு கட்டளை நிரல் எழுதுவோம்.

```
10 PRINT "TIME IN HOURS DISTANCE MOVED"
20 DISTANCE  =  40/4
30 TOTAL  =  0
40 FOR I  =  0.25 TO 1 STEP 0.25
50      TOTAL  = TOTAL + DISTANCE
60       PRINT I, TOTAL
70 NEXT I
100 DISTANCE  =  50 / 4
110 FOR I  =  1.25 TO 2 STEP 0.25
120      TOTAL  =  TOTAL  +  DISTANCE
130       PRINT I,  TOTAL
140 NEXT I
150 END
```

ஒரு இயந்திர பாகத்தை வடிவமைப்பதில் கீழ்க்காணும் சமன்பாடு பயன்படுகிறது.

$$L = \cos(\sqrt{x}) + \tan^{-1}(x^2 + y) + 2.3 \sin(5 + y + 4x) + \log(x + y^4)$$

இதில் x என்பது 1 முதல் 0 வரை 0.1 என்ற வித்தியாசத்திலும், y என்பது 4 முதல் 1 வரை 0.5 என்ற வித்தியாசத்திலும் மாறும்போது, L என்பதன் மதிப்பு என்னவாகும் என்று பட்டியலிடு. இந்த மதிப்புகளில் எது சிறியது என்பதையும் வெளியிடு.

```
10 L = COS ( SQR ( 1 ) +  ATAN ( 1  +  4 )  +  2 .3   *   SIN (5 + 4 + 4) +
LOG (1 + 4 ↑ 4)
20 XMIN = 1 : YMIN = 4 : MIN  =  L
30 REM THE ABOVE ARE INITIAL VALUES
```

```
40 FOR X = 1 TO 0 STEP -0.1
50        FOR Y = 4 TO 1 STEP -0.5
60              L = COS (SQR(X)) + ATAN (X * X + Y)
                +2.3 *SIN (5 + Y + 4 * X) + LOG (X + Y↑4)
70              IF L > = MIN THEN 120
80 REM HERE L < MIN
90             MIN = L
100            XMIN = X : YMIN = Y
110 REM
120       NEXT Y
130 NEXT X
140 PRINT "MINIMUM VALUE    AT X    AT Y"
150 PRINT MIN, XMIN, YMIN
160 END
```

இங்கு ஆரம்பத்தில் குறைந்த மதிப்பாக X = 1, Y = 4 என்பதற்குக் கிடைக்கும் மதிப்பை வைத்துள்ளோம். இந்த X, Y மதிப்புகளையும் நினைவில் வைத்துக் கொள்கிறோம். MIN = 0 என்று வைத்துக்கொள்ளக் கூடாது. ஏன்?

இந்தக் கட்டளை நிரலில் ஏதேனும் தவறு இருக்கிறதா என்று பாருங்கள்? X, Y, L என்பவற்றின் மதிப்புகளைக் கணக்கிட்டோம். ஆனால், பட்டியலாக வெளியீடு செய்ய மறந்துவிட்டோம். (பல மாணவர்கள் தேர்வில் செய்யும் தவறு இது. சரியாகக் கணிப்பார்கள். ஆனால் வெளியீடு செய்ய மறந்துபோவார்கள்). தலைப்புடன் சரியாக வெளியீடு செய்ய கீழே உள்ள கட்டளைகளைச் சேர்த்துக் கொள்ளவேண்டும்.

```
25 PRINT   "X          Y          L"
65         PRINT            X, Y, L
```

இந்தக் கட்டளை எண்கள் சரிதானா என்று பார். 25 என்பதற்குப் பதிலாக 45 அல்லது 55 என்று இருந்தால் தலைப்பு எத்தனை முறை அச்சிடப்படும்? 65 என்பதற்குப் பதிலாக 105 அல்லது 125 என்று இருக்கலாமா?

நினைவில் நிற்க

1. நிச்சயித்துப் பலமுறை செய்வதற்கான பேசிக் கட்டளை FOR.

2. இதைப் பயன்படுத்தும்போது பல விதிகளை நினைவில் வைத்துக் கொள்ள வேண்டும். அவற்றில் முக்கியமான சில -

அ. உள்ளேயிருந்து வெளியே செல்லலாம். ஆனால் உடற்பகுதிக்கு FOR கட்டளை மூலம் மட்டுமே செல்லலாம்.

ஆ. எண்ணும் மாறியைப் பயன்படுத்தலாம். ஆனால் மாற்றக் கூடாது.

இ. கூட்டும் தொகையைக் கொடுக்காமல் இருந்தால் அது 1 என்று எடுத்துக் கொள்ளப்படும். கூட்டும்தொகை எதிர்மறையாகவும் இருக்கலாம். அப்போது எண்ணும்மாறி >= இறுதி மதிப்பா என்று சரி பார்க்கப்படும்.

ஈ. FOR வாக்கியங்கள் ஒன்றுக்குள் ஒன்று முழுவதுமாக இருக்கலாம்.

பயிற்சி வினாக்கள்

1. FOR வளையம் செயல்படும் விதத்தை ஒரு வழிப்படம் மூலம் விளக்கு.
2. FOR வளையத்தின் இலக்கணத்தை விவரி.
3. FOR வளையத்தைப் பயன்படுத்துவதற்கான் விதிமுறைகளில் முக்கியமானவை ஐந்தினைக் கூறு.
4. FOR வளையத்தில் கூட்டும் மதிப்பு சுழி அல்லது எதிர்மறையாக இருந்தால், இந்த வளையத்தைச் செயல்படுத்துவதில் என்ன மாற்றங்கள் ஏற்படுகின்றன?
5. கீழே உள்ளவற்றில் தவறுகள் எவையேனும் இருந்தால் கண்டுபிடி.

அ.
```
10 FOR A$ = 1 TO 5
20        PRINT A$
30 NEXT A$
```

ஆ.
```
10 FOR M = 1 TO 10
20        X = X+5
30        M = M+1
40 NEXT M
```

இ.
```
10 FOR M = 7 TO M = 10
20        X = X
30 NEXT
```

ஈ.
```
10 FOR 17 = 50  TO  5  STEP - 2
20        IF 17 * S < B  THEN  70
30 NEXT I
40 IF X < Y THEN 20
70 J = J + 1
80 GOTO 30
```

6. FOR மற்றும் WHILE வாக்கியங்களுக்கு இடையே உள்ள இரண்டு முக்கிய வேற்றுமைகள் என்ன?

7. 1 முதல் கொடுக்கும் எண் வரை உள்ள எண்களின் மூன்றாம் அடுக்குகளின் கூட்டுத்தொகையைக் கண்டுபிடிக்க ஒரு கட்டளை நிரல் எழுது. கொடுக்கும் எண் 0 அல்லது அதற்குக் குறைவாக இருந்தால் தவறு என்று கூறி, செயல்பாட்டை நிறுத்தி விடு.

8. ஓர் ஆரம்ப எண் (a), வித்தியாசம் (d), மற்றும் எத்தனை எண் (n) என்பவற்றைப் படித்து a, a + d, a + 2d,... a + (n-1) d என்பவற்றை வெளியிடு.

9. BRING ME 1 FLOWERS. I SHALL GIVE 100 RUPEES

BRING ME 2 FLOWERS. I SHALL GIVE 99 RUPEES

BRING ME 100 FLOWERS, I SHALL GIVE 1 RUPEES

என்று நூறு வரிகள் வெளியிட ஒரு கட்டளை நிரல் எழுது.

10. மேலே உள்ள கணக்கில், முதல் வரியில் FLOWER என்றும், கடைசி வரியில் RUPEE என்றும் வெளியிட, கட்டளை நிரலில் என்னென்ன மாற்றங்கள் செய்ய வேண்டும்?

4.12 கட்டளை நிரல்களைச் செயல்படுத்தும் வழி

இதுவரை பேசிக் மொழியின் அடிப்படைத் தகவல் அமைப்புகளையும், எல்லாக் கட்டுப்பாட்டு அமைப்புகளையும் பார்த்துவிட்டோம். அவற்றைக் கொண்டு பல சிறிய கட்டளை நிரல்களையும் எழுதிப் பழகிவிட்டோம். மொழியின் மற்ற அமைப்புகளைப் பார்ப்பதற்கு முன்னால், கட்டளை நிரல்களை ஒரு கணிப்பொறியில் எவ்வாறு செயல்படுத்துவது என்பதை இந்தப் பாடத்தில் பார்ப்போம்.

கணிப்பொறியானது இயக்க முறையின் ஆளுமைக்குக் கீழ் இருக்கும்போது நம்மிடம் உள்ள பேசிக் அமைப்பைச் (GWBASIC, BASICA போன்றவற்றை) செயலாக்க வேண்டும். இதற்கு அந்தப் பெயரைத் திரையில் எழுதி Return என்ற சாவியைத் தட்ட வேண்டும். இந்த பேசிக் அமைப்பு செயல்படும்போது OK என்று வரும். இப்போது கட்டளை நிரலை விசைப்பலகை மூலம் எழுதி முடிக்க வேண்டும். முடிந்ததும் RUN என்று கட்டளையிட்டால், இந்தக் கட்டளை நிரல் செயலாக்கப்படும். கட்டளை நிரலில் இலக்கணப்பிழை எதுவும் இல்லை யென்றால், விடை கிடைக்கும். இந்த விடை கட்டளை நிரலைப் பொறுத்தது. நிரல் சரியாக இருந்தால் சரியான விடை கிடைக்கும். தவறான நிரல், தவறான விடையைக் கொடுக்கும். இதை Garbage in, Garbage out என்பார்கள்.

கட்டளை நிரலில் ஏதாவது ஒரு வரியில் இலக்கணப் பிழை இருந்தாலும், அந்த வரி வரை செயல்பட்டுவிட்டு, தவறிருக்கும் வரியைக் குறிப்பிட்டுக் கூறிவிட்டு நின்றுவிடும். இந்தப் பிழையைத் திருத்திவிட்டு மீண்டும் RUN என்று கட்டளையிட வேண்டும். இலக்கணப் பிழைகளைத் திருத்துவது எளிது. ஆனால், கட்டளை நிரலில் தர்க்கரீதியான பிழைகள் (Logical Errors) இருந்தால் அவற்றைக் கண்டுபிடிப்பது சற்றுக் கடினம். ஏனென்றால் கட்டளை நிரல் செயல்பட்டு விடை கிடைக்கும். ஆனால் அது சரியான விடையாக இருக்காது.

கட்டளை நிரலைத் திரும்பப் பார்க்க வேண்டுமென்றால், LIST என்று கட்டளை இடவேண்டும். கட்டளை நிரல் பெரிதாக இருந்தால், குறிப்பிட்ட சில வரிகளை மட்டும் பார்க்கலாம். உதாரணங்கள் :

LIST	முழுக் கட்டளை நிரலையும் காண்பி.
LIST 25-45	25 முதல் 45 வரை கட்டளை எண்கள் உள்ள வரிகளை மட்டும் காண்பி.
LIST 25	கட்டளை எண்கள் 25 முதல் உள்ள வரிகளை, நிரலின் கடைசி வரை காண்பி.
LIST-400	ஆரம்பம் முதல் 400 ஆம் கட்டளை எண் வரை காண்பி.

எழுதிய கட்டளை நிரலை, பிற்கால உபயோகத்திற்காக, வெளி நினைவகங்களில் ஒரு கோப்பாக எழுதி வைக்கலாம். இதற்கான கட்டளை SAVE. இப்படிக் கட்டளையிட்டால் கணிப்பொறியானது, கோப்பிற்கு ஒரு பெயர் கேட்கும். இந்தப் பெயரில் அதிகபட்சம் 8 + 3 எழுத்துகள் இருக்கலாம். கடைசி 3 எழுத்துகளுக்கு **விரிவு** (Extension) என்று பெயர். இந்தப் பெயர்களில் சிறப்புக் குறியீடுகளைத் தவிர்ப்பது நல்லது. உதாரணங்கள் : FIRST.BAS, MAX.DAT, MASTER93.BAS, SALARY.PAS, இங்கு BAS என்ற விரிவு அது ஒரு பேசிக் மொழிக் கட்டளை நிரல் உள்ள கோப்பு என்பதைக் குறிக்கிறது.

ஏற்கனவே சேமித்து வைத்த கட்டளை நிரலை எடுத்து வந்து அதைச் செயலாக்க முடியும். இதற்கு, முதலில் அந்த நிரலைக் கணிப்பொறிக்குள் கொண்டு வரவேண்டும். இதற்கு LOAD எஎற கட்டளையுடன் கோப்பின் பெயரைக் கொடுக்க வேண்டும். உதாரணங்கள்:

```
LOAD "A : SALES93.BAS"
LOAD "C: CIRCLE"
LOAD "MAXRUN.BAS"
```

LOAD மூலம் உள்ளீடு செய்யும்போது மட்டும். BAS என்று விரிவைக் கூற வேண்டிய அவசியமில்லை. ஏனென்றால் ஒரு பேசிக் கட்டளை நிரலை உள்ளீடு செய்கிறோம் என்பது பேசிக் அமைப்பிற்குத் தெரியும்.

A:,C: போன்றவை எந்தத் தகடு இயக்கி என்பதைக் குறிக்கின்றன.

திரையிலுள்ளதை அழித்துச் சுத்தமாக்க உதவும் கட்டளை CLS. அதாவது Clear Screen என்பதன் சுருக்கம். அழித்தவுடன் **சுட்டி** (Cursor) முதல் வரியில் முதல் இடத்தில் போய் நிற்கும்.

நாம் வரிகளுக்குச் சாதாரணமாக 10, 20, 30 என்று கட்டளை எண்கள் கொடுப்போம். இந்த எண்களைக் கணிப்பொறியையே கொடுக்கச் சொல்லலாம். AUTO என்ற கட்டளை இதைச் செய்யும். உதாரணங்கள் :

AUTO	ஆரம்ப எண் 10, கூடுதல் 10
AUTO 100, 20	ஆரம்ப எண் 100, கூடுதல் 20
AUTO 10, 30	ஆரம்ப எண் 10, கூடுதல் 30.

இப்படித் தானாக எண்கள் கொடுப்பதை நிறுத்த, CONTROL மற்றும் BREAK என்ற இரண்டு சாவிகளையும் ஒரே சமயத்தில் அழுத்த வேண்டும்.

வெளி நினைவகத்தில் உள்ள ஒரு கோப்பினை அதிலிருந்து நீக்க வேண்டுமானால், நாம் DEL என்ற கட்டளையை இயக்க முறைக்குக் கொடுப்போம். பேசிக் அமைப்பிற்கு உள்ளே இருந்து கொண்டே இதைச் செய்ய முடியும். KILL "A:SAL.BAS" என்பது A என்னும் நெகிழ்தகடு இயக்கியில் உள்ள

SAL.BAS என்னும் கோப்பினை அழிக்கும். இங்கு இரண்டு செய்திகள் நினைவில் இருக்க வேண்டும். ஒன்று கோப்பின் முழுப்பெயரையும் கொடுக்க வேண்டும். அழிக்கும்போது ஜாக்கிரதையாக இருக்க வேண்டும் என்பதால், பேசிக் அமைப்பு BAS போன்ற எந்த விரிவையும் தானாகச் சேர்க்காது. இரண்டு - அழிக்கச் சொல்லும் கோப்பு திறக்கப்பட்டு பயன்படுத்திக் கொண்டிருக்கப்பட்டால், ''கோப்பு திறந்திருக்கிறது, தவறு'' என்று சொல்லி அழிக்காமல் நின்றுவிடும்.

தற்போது திரையில் எழுதி வைத்துள்ள கட்டளை நிரலை நீக்கிவிட்டு, ஒரு புது நிரல் எழுத வேண்டுமென்றால் அதற்கு NEW என்ற கட்டளை இடவேண்டும்.

இந்தப் பாடத்தில் நாம் பார்த்த கட்டளைகள் பேசிக் அமைப்பிற்கு இடப் படுபவை. அதனால் இவை **அமைப்பின் கட்டளைகள்** (System Commands) எனப்படும்.

நினைவில் நிற்க

1. கட்டளை நிரல்களைச் செயலாக்கும் பேசிக் அமைப்பின் கட்டளைகளில் சில LOAD, SAVE, RUN, LIST, AUTO, CLS, KILL, NEW.

பயிற்சி வினாக்கள்

1. கீழ்க்காணும் அமைப்புக் கட்டளைகளின் செயல்பாட்டை விளக்கு.

அ. LOAD	ஆ. SAVE	இ. RUN	ஈ. LIST
உ. CLS	ஊ. AUTO	எ. KILL	ஏ. NEW

வரிசைகள், துணைமுறைகள், சார்புகள்

5

பேசிக் மொழியில் இரண்டாம் நிலை என்று கூறத்தக்க செய்திகளை இந்தப் பகுதியில் பார்ப்போம். முதற் பகுதியில் அடிப்படைத் தகவல் வகைகள், அடிப் படைக் கட்டளைகள் மற்றும் அடிப்படைக் கட்டுப்பாட்டு அமைப்புகளைப் பார்த் தோம். இங்கு ஒரு படி மேலே சென்று, பல தகவல்களின் கோவைகளைத் திரட்டி வைத்து, அவற்றைப் பயன்படுத்தும் முறையை முதலில் பார்ப்போம். அடுத்தது, சிக்கலான செயல்பாடுகள் பல இருந்தால் அவற்றை எவ்வாறு ஒன்று திரட்டி வைத்துப் பயன்படுத்துவது என்பதைப் பார்ப்போம். எண்மானக் கோவைகள் ஒரு மதிப்பைக் கணக்கிட உதவுகின்றன. இம்மாதிரிக் கணிப்புகள் பலவற்றைச் செய்து ஒரு மதிப்பைக் கணக்கிட வேண்டியிருக்கும்போது, இதை எண்மானக் கோவை களின் பொதுவடிவம் என்று நினைக்கலாம். சார்புகள் எனப்படும் இவை பற்றி அடுத்ததாகப் படிப்போம். அதாவது, தகவல், கட்டுப்பாடு, கோவை இவற்றின் பெரிய பொது வடிவங்களை இங்குப் பார்க்கப் போகிறோம். அவை முறையே வரிசை, துணைமுறை, சார்பு எனப்படும்.

மேலும், வெளியீடுகளை எவ்வாறு மேம்படுத்தி, முறைப்படுத்துவது என்பதை யும், சரங்களைப் பயன்படுத்தும் விதங்களையும் இப்பகுதியில் பார்ப்போம்.

5.1 வரிசைகளும் அணிகளும்

சென்ற பகுதி பேசிக் மொழிக்கு ஒரு அறிமுகம். அதனால் அடிப்படைத் தகவல்களையும், அடிப்படைக் கட்டுப்பாட்டு அமைப்புகளையும் வைத்து, சிறு கட்டளை நிரல்களை எழுதுவது எவ்வாறு என்று பார்த்தோம். கணிப்பொறியானது கோடிக்கணக்கான தகவல்களை, சிக்கலான கட்டுப்பாடுகள் மூலம் பலவித கணிப்புகளைச் செய்து அலசி ஆராயும் தன்மை படைத்தது. இதனைக் கருத்தில் கொண்டு, இந்தப் பகுதியில் தகவல்களை, கட்டுப்பாடுகளை, மற்றும் கணிப்பு களை, திரட்டி வைக்கும் முறைகளைப் பார்ப்போம். தகவல்களைத் திரட்டி வைக்கும் முறைகளை இந்தப் பாடத்தில் பார்ப்போம்.

ஒரே மாதிரியானதும், ஒன்றுக்கொன்று தொடர்புடையதுமான பல தகவல்கள் இருக்கும்போது, இவற்றிற்கெல்லாம் சேர்த்து, பொதுவாக ஒரு பெயர் கொடுப்பது வழக்கம். உதாரணமாக ஒரு கடையில் ஒரு நாளைய விற்பனை என்பது ஓர் எண். ஒரு மாதத்தின் எல்லா நாட்களின் விற்பனையையும் ஒவ்வொன்றாகப் பார்த்தால், மொத்தம் 31 எண்கள் கிடைக்கும். அவற்றில் சில 0 என்றிருக்கலாம். இந்த 31 எண்களுக்கும் பொதுவான பெயர் ''ஒரு நாள் விற்பனை'' எனலாம்.

இந்த 31 எண்களையும் கணிப்பொறியின் நினைவில், மாறிகளின் மதிப்பாக வைக்க வேண்டுமானால் அவற்றுக்கு எவ்வாறு பெயரிடுவது? இதேபோல் கோடிக்கணக்கான எண்களை நினைவில் வைக்க வேண்டியிருக்கும். அதனால், ஒவ்வொன்றுக்கும் ஒரு தனிப்பெயர் கொடுத்தால் கட்டுபடியாகாது. எல்லா வற்றிற்கும் பொதுவாக உள்ள பெயரைப் பயன்படுத்த வேண்டும் என்பது தெளி வாகத் தெரிகிறது. நம் உதாரணத்தில், விற்பனை என்ற பொதுப் பெயரைப் பயன் படுத்தலாம். முதல் நாள் விற்பனையை விற்பனை (1) என்றும், இரண்டாம் நாள் விற்பனையை விற்பனை (2) என்றும் குறிப்பிடலாம். ஒரு பொதுப்பெயர் கொண்டுவரும், ஒரே மாதிரியான தகவல்கள் **வரிசை** (Array) எனப்படும். இங்கு அடைப்புக் குறிக்குள் வரும் முழு எண் **கீழ்க்குறியீடு** (Suffix) எனப்படும். இது அந்த வரிசையில் உள்ள எத்தனையாவது தகவலைப் பயன்படுத்துகிறோம் என்பதைக் குறிப்பிடும். நாம் பார்த்த வரிசையில் ஒரு கீழ்க்குறியீடு வருகிறது. இது **ஒருபரிமாண வரிசை** (One Dimensional Array) எனவும் அழைக்கப்படும்.

ஒரு படி மேலே செல்வோம். ஒரு வருடத்தில் உள்ள எல்லா நாட்களுக்கும் உள்ள விற்பனையை எப்படிக் குறிப்பிடலாம்? ஏற்கனவே பார்த்தபடி விற்பனை (1) முதல் விற்பனை (366) வரை குறிப்பிடலாம். இதில் ஒரு குறை இருக்கிறது. ஆகஸ்ட் 10 இல் விற்பனை என்ன? - என்று கேட்டால் உடனே பதில் சொல்ல முடியாது. ஆகஸ்ட் 10 அந்த வருடத்தில் எத்தனையாவது நாள் என்று கண்டுபிடித்து, அதற்குப் பிறகே விடை சொல்ல முடியும்.

விற்பனை என்பதை மாதம், நாள் என்று இரண்டு கீழ்க்குறியீடுகள் கொண்ட பட்டியலாக, கீழே உள்ளதுபோல் நினைத்தால் கணக்கிடுவது நமக்கு எளிதாக இருக்கும்.

மாதம்	நாள் 1	2...	31
1	10	7...	70
2	6	.	
.	.		
.	.		
12	50	48...	66

இங்கு விற்பனை (மாதம், தேதி) என்பது ஒரு எண்ணைக் குறிக்கும். விற்பனை (8, 10) என்பது 8ஆம் மாதத்தில், 10 ஆம் நாள் விற்பனையைக் குறிக்கும். பட்டியலில் இங்கு மொத்தம் 12 x 31 = 372 தகவல்கள் இருக்கின்றன (ஏப்ரல் 31 போன்று இல்லாத நாட்களிலும், விடுமுறை நாட்களிலும் விற்பனை 0 என்போம்).

இம்மாதிரிப் பட்டியல்களை **இருபரிமாண வரிசை** (Two Dimensional Array) அல்லது **அணி** (Matrix) எனலாம். வரிசை என்ற பொதுப் பெயரில் இருபரிமாண வரிசைகளும் அடங்கும்.

தகவல் ஒவ்வொன்றும் எண்ணாக இருக்கலாம், அல்லது சரமாக இருக்கலாம். ஆனால், ஒரு வரிசையில் இருக்கும் தகவல்கள் எல்லாமே ஒரே மாதிரி இருக்க வேண்டும். சில தகவல்கள் எண்களாகவும், சில தகவல்கள் சரங்களாகவும் இருக்கக் கூடாது. ஒரு வரிசையில் எத்தனைத் தகவல்கள் இருக்கின்றன என்பது தெரிந்தால் தான், அத்தனைக்கும் தேவையான இடங்களைக் கணிப்பொறியினால் ஒதுக்கி வைக்க முடியும். அதனால், இந்த விவரத்தைக் கட்டளை நிரலின் ஆரம்பத்திலேயே கொடுப்பது அவசியம். இது **பரிமாண வாக்கியம்** (Dimension Statement) எனப்படும். நாம் பார்த்த உதாரணத்திற்கு, இருபரிமாண வரிசைக்கு எழுத வேண்டிய பரிமாண வாக்கியம்.

DIM SALES (12, 31)

இதைப் பார்த்தவுடன் மொழிமாற்றி எத்தனை இடங்களை ஒதுக்கும்? 12 x 31? இல்லை! 13 x 32 இடங்களை ஒதுக்கும். ஏன்? பேசிக்கில் கீழ்க் குறியீடு கள் சுழியிலிருந்து ஆரம்பிக்கும். 0, 1, 2,.......12 என்று 13, 0, 1..., 32 என்று 33, ஆக மொத்தம் 13 x 32 இடங்கள். (சில மொழிமாற்றிகள் கீழ்க் குறியீட்டை ஒன்றிலிருந்து ஆரம்பிக்கலாம்). நமக்கு எளிமையாக இருக்க வேண்டும் என்பதற்காகச் சில இடங்களை விரயம் செய்வதில் தவறில்லை. அதனால், கீழ்க்குறியீடு சுழி என்று வரும் தகவல்களுக்கான இடங்களைப் பயன்படுத்தாமல் விட்டுவிடலாம்.

கீழ்க்குறியீடு பெற்ற மாறிகளை வரையறை செய்வதிலும், அவற்றைப் பயன்படுத்துவதிலும் சில விதிமுறைகள் உண்டு. அவற்றை ஒவ்வொன்றாகப் பார்ப்போம்.

1. ஒரு வரிசையில் அதிக அளவு 255 பரிமாணங்கள் இருக்கலாம். ஆனால், எந்த வாக்கியமும் ஒரு வரிக்குள் அடங்க வேண்டும் என்பதால், உண்மையில் குறைந்த அளவு பரிமாணங்களையே பயன்படுத்த முடியும். பொதுவாக, பரிமாணங்கள் 1,2,3 உள்ள வரிசைகளே பெரிதும் பயன்படுத்தப்படுகின்றன. மேலே பார்த்த உதாரணத்தில், 10 ஆண்டுகளுக்கான விற்பனையை, வருடம், மாதம், நாள் என்று மூன்று கீழ்க்குறியீடுகள் கொண்ட முப்பரிமாண வரிசையாகக் குறிப்பிடலாம்.

2. எந்த ஒரு பரிமாணத்திலும் அதிகபட்சம் 32,767 தகவல்கள் இருக்கலாம். அதாவது ஒரு கீழ்க்குறியீடு 0 முதல் 32,765 வரை இருக்கலாம். ஒரு 10 பரிமாண வரிசையில் ஒவ்வொன்றிலும் இத்தனை கீழ்க்குறியீடுகள் வர முடியுமா? எவ்வளவு பெரிய வரிசைகளை உருவாக்கி, பயன்படுத்த முடியும் என்பது, கணிப்பொறியில் உள்ள நினைவகத்தின் அளவைப் பொறுத்தது.

3. எந்த ஒரு பரிமாணத்திலாவது 10க்கு மேற்பட்ட எண்ணை கீழ்க்குறியீடாகப் பயன்படுத்தினால், கட்டாயம் அந்த மாறியை வரையறை செய்ய, ஒரு பரிமாண வாக்கியம் தேவை. எல்லாப் பரிமாணங்களிலும் 10 அல்லது அதற்குக் குறைவான எண்களை மட்டுமே கீழ்க்குறியீடுகளாகப் பயன்படுத்தினால், அந்த மாறிக்கு பரிமாண வாக்கியம் கொடுக்க வேண்டிய கட்டாயம் இல்லை. ஏனென்றால் இம்மாதிரி இடங்களில் கணிப்பொறியே 11 அல்லது 11 × 11, 11 × 11 × 11 போன்று, பரிமாண எண்ணிக்கையைப் பொறுத்து, இடங்களைத் தானாக ஒதுக்கிவிடும்.

DIM TEST (10), MARKS (6, 9), NAME$ (4, 7, 10)

என்ற வாக்கியம் தேவையில்லை. ஆனால், நாம் பயன்படுத்தும் எல்லா வரிசைகளுக்கும் பரிமாண வாக்கியம் எழுதுவது ஒரு நல்ல பழக்கம்.

4. ஒரு பரிமாண வாக்கியத்தில் ஒன்றுக்கு மேற்பட்ட வரிசைகளைக் குறிப்பிடலாம். உதாரணம்.

DIM A (20, 3), NAMES$ (50), RUNS (500), TEMP$ (100, 200)

இங்கு NAME$ என்பது சரங்களின் வரிசையைக் குறிக்கும்.

5. கீழ்க்குறியீடாகப் பயன்படுத்தப்படும் எண் ஒரு முழு எண்ணாக இருப்பது நல்லது. ஆனால் அவசியமில்லை. பின்னமாக இருந்தால், அதை அருகிலுள்ள முழு எண்ணாக மாற்றி எடுத்துக்கொள்ளும். உதாரணமாக A (2.3) என்பதும் A(1.8) என்பதும் A(2) என்பதைக் குறிக்கும்.

6. வரிசைகளைப் பயன்படுத்தும்போது, கணிப்பொறி தானாகவே, வரிசையிலுள்ள எண்களுக்கு 0 என்றும், சரங்களுக்கு வெற்றுச் சரங்களாகவும், முதலில் தொடக்க மதிப்புகளைக் கொடுத்துவிடும்.

வரிசைகளில் எத்தனை தகவல்கள் உள்ளன என்பது நமக்கு முன்கூட்டியே தெரியும். தகவல்கள் ஒவ்வொன்றுக்கும் ஒரே மாதிரியான செயல்பாடுகளைச் செய்ய வேண்டியிருக்கலாம். இதற்கு FOR கட்டளை பெரிதும் பயன்படும். இரு பரிமாண வரிசை என்றால் ஒன்றுக்குள் ஒன்றாக இரண்டு FORகளைப் பயன்படுத்த வேண்டும். வரிசைகளில் உள்ள கீழ்க்குறியீடுகளுக்கும், FOR வாக்கியத்தில் வரும் எண்ணும் மாறிக்கும் நெருங்கிய தொடர்பு உண்டு. எண்ணும் மாறியை கீழ்க் குறியீடாகப் பயன்படுத்துவது ஒரு சிறந்த, எளிய, வசதியான வழி. எடுத்துக்காட்டு களைப் பார்ப்பதன் மூலம் வரிசைகளைப் பயன்படுத்தும் விதத்தை அறிந்து கொள்வோம்.

1. ஒரு மாதத்தில் ஒவ்வொரு நாள் விற்பனையையும் படி. அம்மாதத்தின் மொத்த விற்பனையைக் கணக்கிடு.

எளிதில் புரிந்துகொள்வதற்காக செயல்களைத் தனித்தனியாகச் செய்வோம்.

```
10 DIM SALES (31)
20 FOR I = 1 TO 31
30         PRINT "GIVE SALES AMOUNT FOR DATE", I
40         INPUT SALES (I)
50 NEXT I
60 REM
70 TOTAL = 0
80 FOR I = 1 TO 31
90         TOTAL = TOTAL + SALES(I)
100 NEXT I
110 PRINT "TOTAL SALES=", TOTAL
120 END
```

கீழ்க்குறியீடுகளைப் பயன்படுத்தும்போது, குறிப்பிடப்பட்ட எல்லைக்கு மேலே அல்லது கீழே சென்றால் என்ன ஆகும்? உதாரணமாக மேலே உள்ள கட்டளை நிரலில், தவறுதலாக

```
20 FOR I = 1 TO 32
```

என்று எழுதப்பட்டால், 40ஆம் வரியில் INPUT SALES (32) என்று ஒரு முறை வரும். அப்படிப்பட்ட இடங்களில் கணிப்பொறி **எல்லைக்கு வெளியே** (Out of Range) என்ற பிழைச் செய்தியைக் கொடுத்துவிட்டு செயல்பாட்டை நிறுத்திவிடும்.

2. நூறு எண்களைப் படித்து, அவற்றை ஏறு வரிசையில் எழுது.

ஏறு வரிசை அல்லது இறங்கு வரிசையில் ஒழுங்குபடுத்தி வைப்பது என்பது கணிப்பொறி இயலில் ஒரு முக்கியமான செயல். கணிப்பொறி செய்யும் செயல் களில் சுமார் மூன்றில் ஒரு பங்கு இம்மாதிரி ஒழுங்குபடுத்தி வைத்துக் கொள்வது

தான் என்று கணக்கிடப்பட்டிருக்கிறது! ஏறு வரிசையில் (அல்லது இறங்கு வரிசையில்) எழுத பல செயல் முறைகள் உள்ளன. எளிதாகப் புரியக்கூடிய ஒன்றை இங்கு பார்ப்போம். முதலில் போலிக் குறியீட்டில் எழுதலாம்.

1. *100 எண்களைப் படி. அவை* A (1)...A (100).

2. I *என்பது 1 முதல் 99 வரை செய்*
   (A (I) *முதல்* A (100) *வரை உள்ளவற்றில் சிறிய எண்ணைக் கண்டுபிடி. அது* A (POS) *என்போம்.*
   A (I) *ஐயும்* A (POS) *ஐயும் இடம் மாற்று.*
)

3. *வரிசையை வெளியிடு.*

இதைப் புரிந்துகொண்டு, அதாவது எப்படி இந்தச் செயல்முறை கொடுத்த எண்களை ஏறுவரிசைக்கு மாற்றுகிறது என்பதை, உலா வருவதன் மூலம் சரிபார்த்த பிறகு, அடுத்துவரும் கட்டளை நிரலைப்படி.

```
10 DIM A (100)
20 FOR I = 1 TO 100
30        PRINT " GIVE THE", I, "TH NUMBER"
40        INPUT A(I)
50 NEXT I
60 REM
70 FOR I = 1 TO 99
80        MIN = A (I) : POS = I
90        FOR J = I + 1 TO 100
100             IF A ( J ) < MIN THEN MIN = A ( J ) : POS = J
110   REM ELSE NOTHING TO DO
120       NEXT J
130 REM
140 REM MIN POSITION FOUND. INTERCHANGE THEM
150       TEMP = A ( I ) : A ( I ) = A ( POS ) : A ( POS ) = TEMP
160 NEXT I
170 REM
200 FOR I = 1 TO 100
210       PRINT A ( I )
220 NEXT I
230 END
```

3. ஒரு கல்லூரி 1989இல் தொடங்கப்பட்டது. அதில் நான்கு வகுப்புகள் உள்ளன. ஒவ்வொரு வகுப்பிற்கும், ஒவ்வொரு வருடமும் ஒரு மாணவர் தலைவர் உண்டு. இவர்களின் பெயர்கள் அடங்கிய பட்டியல் கீழே உள்ளது. இந்தப் பட்டியலை கணிப்பொறியின் நினைவில் வை. பிறகு, ஒரு பெயரைப் படித்து, அந்த மாணவன்/மாணவி எந்த வகுப்பிற்கு, எந்த ஆண்டு தலைமைப் பொறுப்பு வகித்தான்(ள்) என்று கூறு. பிறகு ஒரு வகுப்பு எண்ணைப் படித்து, அந்த வகுப்புத் தலைவர்கள் பெயர்களை, வருடத்துடன் வெளியிடு.

கணக்கே போலிக் குறியீட்டில் கொடுத்தது போல் இருக்கிறது. இதில் உள்ளதை வரிசையாகச் செயல்படுத்துவோம். கணக்கைச் செயல்படுத்தத் தேவையான பட்டியல் கீழே உள்ளது.

	ஆண்டு		
வகுப்பு	*1989*	*1990*	*1991*
1	ஆனந்தி	கண்ணன்	ராதா
2	குமரன்	மணி	சிவாஜி
3	சுபாஷ்	ரவி	சந்திரா
4	நிலா	சூர்யா	பாலன்

தகவல்களைப் பார்க்கும்போது, அவற்றை NAME$ என்ற பெயரில் இரு பரிமாண வரிசையாக, அதாவது அணியாக வைப்பது சிறந்தது எனத் தெரிகிறது. ஒரு கட்டளை நிரலை எழுத ஆரம்பிப்பதற்கு முன், அதில் வரும் தகவல்களை ஆராய்ந்து அவற்றை எந்த விதங்களில் வைக்கலாம் என்பதை, அதாவது தகவல் களின் அமைப்பை முடிவு செய்ய வேண்டும். தகவல்களை சரியாகப் புரிந்து கொண்டால், அவற்றை என்ன செய்ய வேண்டும் என்பது தெளிவாகப் புரியும்.

```
10 DIM NAME$ ( 4, 3 )
20 REM NAME ( CLASS NO, YEAR )
30 FOR I  =  1 TO 4
40      FOR J  =  1989 TO 1991
50           PRINT "GIVE NUMBER CLASS = ",  I,
60           PRINT "YEAR = ",  J
70           INPUT NAME$ ( I,  J - 1988 )
80      NEXT J
90  NEXT I
100 REM NOTE SUFFIX IS J - 1988.
```

```
110 PRINT  " GIVE A NAME "
120 INPUT FINDNAME$
130 FOR I  =  1 TO 4
140     FOR J  =  1 TO 3
150          IF FINDNAME$ < > NAME$ ( I, J ) THEN 200
160          REM ELSE - HERE EQUAL NAMES
170          PRINT FINDNAME$, "IS LEADER OF"
180          PRINT I, "TH CLASS"
190          PRINT "IN THE YEAR" , J  +  1988
200     NEXT J
210 NEXT I
220 REM
230 PRINT "GIVE CLASS NUMBER"
240 INPUT N
250 PRINT "NAME      YEAR"
260 FOR K = 1989 TO 1991
270     PRINT NAME$ ( N, K - 1988 ), K
280 NEXT K
290 END
```

முப்பரிமாண வரிசை ஒன்றைப் பயன்படுத்தும் உதாரணத்தைப் பார்ப்போம்.

4. ஒரு மாநிலத்தில் ஒவ்வொரு வருடமும் வரவு செலவை திட்டமிடும்போது, திட்டமதிப்பு, திருத்தப்பட்ட திட்ட மதிப்பு, சரியான இறுதி மதிப்பு என்று மூன்று எண்கள் கிடைக்கின்றன. *15* மாநிலங்களில், *8* வருடங்களுக்கு இம்மாதிரியான தகவல்கள் உள்ளன. இதைப் படித்து நினைவில் வைத்துக்கொண்டு, ஒரு குறிப்பிட்டுக் கேட்கும் வருடத்தில், ஒவ்வொரு மாநிலத்திலும் திட்டமதிப்பு எவ்வளவு, சரியான இறுதி மதிப்பு எவ்வளவு என்பதை வெளியிடு.

இங்கு மாநிலங்களை *1* முதல் *15* வரையும், வருடங்களை *1* முதல் *8* வரையும், மதிப்பை *1* முதல் *3* வரையும் எடுத்துக்கொள்ளலாம்.

```
10 DIM BUDGET (15, 8, 3)
20 FOR I = 1 TO 15
30          FOR J = 1 TO 8
40                 FOR K = 1 TO 3
```

```
50            PRINT "GIVE BUDGET FOR",  I , J, K
60            INPUT BUDGET (I,J,K)
70         NEXT K
80   NEXT J
90 NEXT I
100 PRINT "GIVE YEAR NUMBER"
110 INPUT YEAR
120 PRINT " STATE          BUDGET          ACTUAL"
130 FOR S = 1 TO 15
140         PRINT S, BUDGET (S, YEAR, 1),
145         PRINT BUDGET (S, YEAR, 3)
150 NEXT S
160 END
```

நினைவில் நிற்க

1. பல தகவல்களை ஒரே பெயரில் குறிப்பிடுவதற்கு வரிசைகள் பயன்படும். இதற்கு எல்லாத் தகவல்களும் ஒரே மாதிரியாக இருக்க வேண்டும்.

2. எத்தனை தகவல்கள் என்பதை DIM வாக்கியத்தின் மூலம் கட்டளைத் தொடரின் ஆரம்பத்தில் கொடுக்க வேண்டும். இந்த வாக்கியம் சில சமயங்களில் அவசியம் இல்லை. ஆனால், கொடுப்பது நல்ல வழக்கம்.

3. FOR தொடரில் வரும் எண்ணும் மாறியை, கீழ்க்குறியீடாகப் பயன்படுத்துவது பொதுவான வழக்கம். இரு அல்லது முப்பரிமாண வரிசைகளை ஆராயும்போது FOR வாக்கியங்கள் ஒன்றுக்குள் ஒன்றாக வரும்.

பயிற்சி வினாக்கள்

1 வரிசைகளையும், அணிகளையும் எவ்வாறு வரையறை செய்வது?

2 கீழ்க்குறியீடுகளாக எவற்றைப் பயன்படுத்தலாம்?

3. ஒரு வரிசையை வரையறை செய்ய வேண்டியது அவசியமா?

4. கீழே உள்ளவற்றில் தவறுகள் எவையேனும் இருந்தால் கண்டுபிடி.

அ.
```
10 DIM A$ (10, 3)
20 INPUT A$ (7)
```
ஆ.
```
1 INPUT AAA$ (23)
```

இ. 1 INPUT AB$ (3)

ஈ. 10 FOR J = 100 TO 3 STEP -1
20 INPUT A(I)
30 NEXT j

உ. 5 DIM M (100)
10 FOR J = 5 TO -5 STEP -1
20 INPUT M(J)
30 NEXT J

ஊ. 100 A(I + J) = B$ (I)

எ. 100 A$ (I) = BB$ (I, J, K, L)

ஏ. 100 NAME (I$) = "GOOD"

ஐ. 100 AA (I) = "MAN"

ஒ. 50 A (I - 2) = 3

ஓ. 60A (7. 8) = 7.8

ஔ.100 A (I, J, K) = A (K, J, I)

5. ஒரு வகுப்பில் 30 மாணவர்கள் உள்ளனர். அவர்களுடைய மதிப்பெண்கள் 6 பாடங்களில் தரப்பட்டுள்ளன. அவற்றை நினைவில் வைத்துக்கொண்டு கீழ்க்காணும் வேலைகளைச் செய்ய வேண்டும்.

அ. ஒவ்வொரு மாணவனும் தேர்வு பெற்றானா என்று பார்க்க வேண்டும். அதற்கு, ஒவ்வொரு பாடத்திலும் குறைந்தது 40 மதிப்பெண்கள் பெற வேண்டும்.

ஆ. தேர்வு பெற்ற மாணவர்களுக்கு மட்டும் மொத்த மதிப்பெண்களைக் கணக்கிட்டு, அதில் அதிக மதிப்பெண்ணைக் கணக்கிடு.

இ. முதல் மதிப்பெண் பெற்ற மாணவர்கள் அனைவரின் முழு விவரங்களையும் வெளியிடு.

6. போகும் வேகத்தைப் பொறுத்து வண்டியின் திறன் (1 லிட்டருக்கு எத்தனை கிலோ மீட்டர்கள் போகும் என்பது) மாறுபடும். இதற்கான பட்டியல் கீழே உள்ளது. இதைப் படித்து ஒவ்வொரு வண்டியும் எந்த வேகத்தில் திறம்பட இயங்கும் என்பதைப் பார்க்க, ஒரு கட்டளை நிரல் எழுது.

வண்டி எண்

வேகம்	*1*	*2*	*3*
20	*40*	*45*	*60*
40	*60*	*50*	*70*
60	*50*	*75*	*80*
80	*40*	*60*	*90*
100	*30*	*40*	*60*

7. ஃபிபனாக்கி வரிசை என்பதில் முதல் இரண்டு எண்கள் 1, 1. அதன் பிறகு வரும் ஒவ்வொரு எண்ணும் முந்தைய இரண்டு எண்களின் கூட்டுத்தொகை. அதாவது மூன்றாம் எண் 1+1= 2; நான்காம் எண் 2+1=3. இந்த வரிசையில் உள்ள முதல் 20 எண்களைக் கணக்கிட்டு வெளியிடு. அவற்றின் கூட்டுத் தொகையைக் கண்டுபிடி.

8. ஒரு வரிசையில் (அதிகபட்சம் 500 எண்கள்) உள்ள எண்களைப் படித்து, அதைத் தலைகீழாக மாற்றி எழுது. உதாரணமாக, 4, 77, 6, 5 என்ற வரிசை 5, 6, 77, 4 என மாறவேண்டும். முதலில், வரிசையில் எத்தனை எண் என்று கேட்டு வாங்கு. பிறகு அத்தனை எண்களைப் படி.

9. ஒரு வரிசையில் எத்தனை எண்கள் பத்துக்கும் நூறுக்கும் இடையில் இருக்கின்றன எனப் பார்க்க ஒரு கட்டளை நிரல் எழுது.

10. 16×16 அளவுள்ள ஒரு பெருக்கல் பட்டியலைத் தயார் செய்து நினைவில் வை. இங்கு MUL (I, J) என்பதில் $I \times J$ என்ற மதிப்பு இருக்க வேண்டும்.

11. நான்கு ஊர்களின் ஒவ்வொரு நாளைய அதிக மற்றும் குறைந்த வெப்ப அளவு ஒரு வாரத்திற்குக் கொடுக்கப்பட்டிருக்கிறது. அந்த வாரத்தில், ஒவ்வொரு நாளிலும் எந்த ஊரில் அதிக வெப்பம், எந்த ஊரில் குறைந்த வெப்பம் என்று கண்டுபிடி.

12. ஒரு நிறுவனத்திற்கு 15 கிளைகள் உள்ளன. ஒவ்வொன்றிலும் 20 விற்பனையாளர்கள் பணிபுரிகின்றனர். இவர்கள் ஒவ்வொருவரும் அவர்களுக்கு ஒதுக்கப்பட்ட இடங்களை 5 தொகுதிகளாகப் பிரித்து பணிபுரிகின்றனர். ஒவ்வொரு தொகுதியிலும் அந்த ஆண்டு எவ்வளவு லட்சம் ரூபாய்களுக்கு விற்பனை செய்திருக்கிறார்கள் என்ற தகவல்கள் உள்ளன. இதை எப்படி நினைவில் வைப்பது என்று கூறு. ஒரு கிளை எண்ணைக் கொடுத்தால், அந்தக் கிளையில் மொத்த விற்பனை எவ்வளவு என்று கண்டுபிடிக்க ஒரு கட்டளை நிரல் எழுது.

13. ஐந்தாம் கணக்கில் ஒரு வகுப்பில் உள்ள மாணவர்கள் மட்டும் எடுத்துக் கொள்ளப்பட்டார்கள். 15 வகுப்பு மாணவர்களை எடுத்துக் கொண்டால், தகவல்களின் அமைப்பையும், பெயர்களையும் எவ்வாறு மாற்ற வேண்டும் என்று கூறு, பிறகு கட்டளை நிரலை மாற்றி எழுது.

5.2 துணைமுறைகள்

ஒரு கட்டளை நிரலில், குறிப்பிட்ட ஒரு துணைப்பணியை, பல இடங்களில் திரும்பத் திரும்ப செய்ய வேண்டியிருக்கலாம். இந்தப் பணிக்கான செயல்பாடுகளை ஓர் இடத்தில் மட்டும் எழுதி வைத்து, அதைத் தேவையானபோதெல்லாம் பயன்படுத்திக் கொள்வதுதான் புத்திசாலித்தனம். ஆனால் இப்படிச் செய்வதில் ஒரு சிக்கல் இருக்கிறது.

ஒரு துணைப் பணிக்கான செயல்பாடுகளை வரிகள் 1000 முதல் 2000 வரை எழுதி வைத்திருப்பதாக நினைப்போம். 100 மற்றும் 200 ஆம் வரிகளில் இந்தப் பணியைச் செய்ய வேண்டும் என்றால் எப்படிச் செய்வது? 100 ஆம் வரியில் GOTO 1000 என எழுதலாம். 2000 வரி முடிந்ததும், அதாவது பணி முடிந்ததும் திரும்பி 110 ஆம் வரிக்கு வர வேண்டும். இதற்காக 2010 ஆம் வரியை GOTO 110 என எழுதலாம். இப்போது 200 ஆம் வரியில் GOTO 1000 என்று எழுதினால், பணி முடிந்ததும் 210க்குத் திரும்ப வேண்டும். ஆனால் 2010 ஆம் வரியில் இருப்பதோ GOTO 110. இதுதான் சிக்கல்.

எங்கேயிருந்து இந்தப் பணியைச் செய்ய கட்டளை வந்தது என்பதைக் கணிப்பொறியே நினைவில் வைத்துக் கொண்டால், பணி முடிந்ததும் தானாகவே இந்த இடத்திற்கே வந்து, இதற்கடுத்து இருக்கும் வாக்கியத்திற்குச் செல்லலாம். இப்படிச் செய்வது கட்டளை நிரல் எழுதுவதை எளிதாக்கும்.

இப்படிச் செய்ய இரண்டினைக் கருத்தில் கொள்ள வேண்டும். ஒன்று - ஒரு துணைப் பணிக்காகத் தொகுத்து எழுதப்படும் வரிகள் தனித்து நிற்கும் ஒரு தொகுதியாக நினைக்கப்பட வேண்டும். இரண்டு - அதற்குள் போவதும், திரும்பி வருவதும் ஒழுங்குப்படுத்தப்பட வேண்டும். குறிப்பிட்ட முறையில்தான் அவ்வாறு போவதும், வருவதும் இருக்க வேண்டும். இவற்றை எப்படிச் செய்வது என்று பார்ப்போம்.

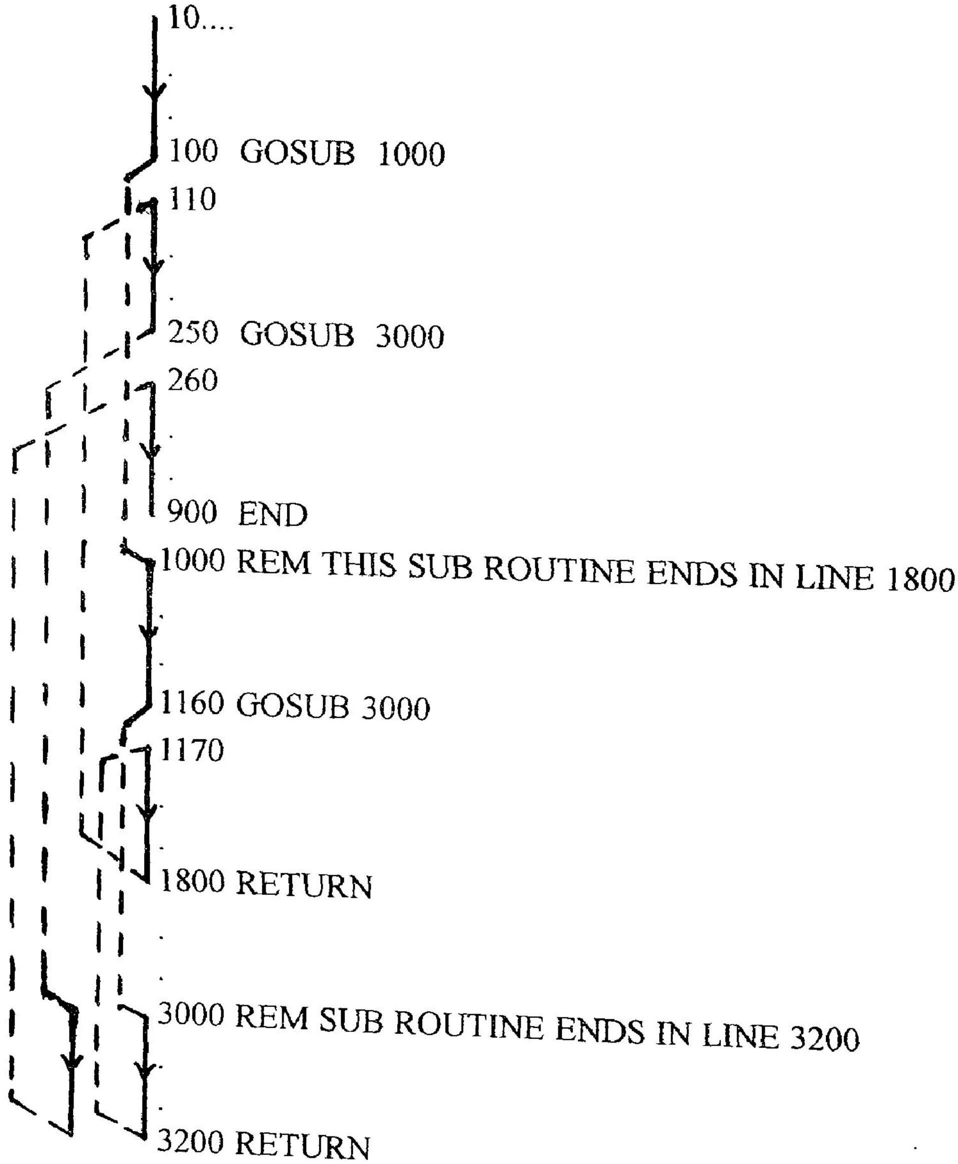
துணைப்பணிக்காக எழுதப்படும் கட்டளைத் தொகுதிக்கு துணைமுறை (Subroutine) என்பது பெயர். துணைமுறைக்கு உள்ளே GOSUB n, அல்லது ON i GOSUB n1,n2,... என்ற இருவிதங்களில் மட்டுமே போகவேண்டும். இங்கு n, n1, n2, போன்றவை, துணை முறை தொடங்கும் வரி எண்களைக் குறிக்கின்றன. இங்கு GOSUB என்பது GOTO போன்று செயல்படும். ஆனால் துணைமுறை செய்து முடிக்கப்பட்டவுடன் திரும்பவும் இதே இடத்திற்குக் கட்டுப்பாடு **தானாக** வந்து சேரும்.

கட்டளை நிரலில் செயல்பாடுகள் முடிந்ததை STOP என்பதன் மூலம் குறிப்பிடுகிறோம். துணைமுறையில் செயல்பாடுகள் முடிந்ததை RETURN என்ற

சொல் குறிப்பிடுகிறது. ஒரு துணைமுறையில் எத்தனை இடங்களில் வேண்டுமாலும் RETURN என்ற சொல் இருக்கலாம். உதாரணம்

```
1000 REM SUBROUTINE TO INCREASE PAY
1010 IF PAY > 1000 GOTO 1100
1020 P = P+100 : RETURN
1100 P = P*1.2  : RETURN
```

ஒரு துணைமுறையிலிருந்து மற்றொரு துணைமுறைக் கூப்பிடும் ஓர் உதாரணத்தில் கட்டுப்பாடு செல்லும் வழியைக் கீழே காணலாம்.



துணைமுறையின் முதல் வரி எதுவாக வேண்டுமானாலும் இருக்கலாம். REM ஆகக் கூட இருக்கலாம்.

துணைமுறையில் பயன்படுத்தப்படும் மாறிகள் எல்லாம், கட்டளை நிரலில் பயன்படுத்தப்படும் அதே மாறிகள்தான். வேறெதுவுமில்லை.

துணைமுறையைக் கட்டளை நிரலின் எந்த இடத்திலும் எழுதலாம். END க்குப் பிறகும் கூட எழுதலாம். இப்படிச் செய்யும் போது, கட்டுப்பாடு நாம் சொல்லாமல் தானாகத் துணைமுறைக்குள் செல்வது எளிதாகத் தடுக்கப்படுகிறது.

துணைமுறையிலிருந்து GOTO, IF, ON...GOTO போன்ற கட்டளைகள் மூலம் வெளியே வரக்கூடாது. அதே போல் உள்ளே செல்லவும் கூடாது.

ஒரு துணைமுறைக்குள் இருக்கும்போது, வேறொரு துணைமுறையைக் கூப்பிடலாம். ஆனால் அதே துணைமுறையைக் கூப்பிடக்கூடாது. ஏனென்றால் இது ஒரு வளையத்தைத் தோற்றுவிக்கும்.

ஒரு துணைமுறை வேறொரு துணைமுறையைக் கூப்பிடும்போதும் ஒரு வளையம் தோன்றக் கூடாது. உதாரணமாக, A என்ற துணைமுறை B ஐயும் B என்பது C ஐயும், C என்பது A ஐயும் கூப்பிடுவதாக வைத்துக் கொண்டால் A-B-C-A என்ற வளையம் தோன்றும். இது கூடாது.

ஒரே ஒரு முறை பயன்படுத்துவதாக இருந்தாலும், பல உத்தரவுகள் சேர்ந்து ஒரு குறிப்பிட்ட பணியைச் செய்வதாக இருந்தால் அதை ஒரு துணைமுறையாக எழுதுதல் நல்லது. இது கட்டளைநிரல் எழுதுவதையும், புரிந்து கொள்வதையும் எளிதாக்கும்.

வழிப்படம் என்றால் என்ன என்று சொல்வது; அடுத்து பேசிக் மொழியின் இலக்கணத்தைக் கூறுவது; சிறு உதாரணங்கள் கொடுப்பது; பிறகு மாணவரை கட்டளை நிரல்கள் எழுதச் சொல்வது-இம்முறையில் பயிலும் மாணவர்கள், துணைமுறை என்றால் கடினமானது, அதைத் தவிர்த்து விட்டு கட்டளை நிரல்கள் எழுதலாம் என நினைப்பது இயற்கை.

இந்தப் புத்தகத்தில், கட்டளை நிரல் எழுதுவதை இரு பகுதிகளாக்கி, முதல் பகுதியில் போலிக் குறியீட்டில் எழுதுகிறோம். அடுத்த பகுதியில் போலிக் குறியீட்டிலிருந்து பேசிக் மொழியில் மாற்றுகிறோம். போலிக் குறியீட்டில் எழுதும்போது, சில துணைச் செயல்முறைகளை, என்ன செய்ய வேண்டும் என்று கூறிவிட்டு, எப்படிச் செய்வது என்று கூறாமல் விட்டுவிடலாம். இவற்றை எப்படிச் செய்வது என்பதைத் தனியாக துணைமுறையாக எழுதலாம். இப்படிச் செய்யும் போது, கணக்கைத் தீர்க்கும் வழியில் அதிகக் கவனம் செலுத்தலாம். சில பகுதிகளை எப்படிச் செய்வது என்பதைப் பிறகு தனியாகத் தீர்மானித்துக் கொள்ளலாம். இதனால், போலிக்குறியீட்டில் தீர்வு முறையை எழுதுவது எளிதாகும். இதைக் கருத்தில் கொண்டு மாணவர்கள் துணைமுறைகளை அச்சமின்றி தாராளமாகப் பயன்படுத்த முன் வர வேண்டும். கல்விச்சாலைக்கு வெளியே வந்தவுடன் எழுதப்படும் கட்டளை நிரல்கள் பெரியதாக இருக்கும். துணை முறைகள் இல்லாமல் இவற்றை

எழுதுவது இயலாத காரியம். மேலும் ஸி மொழியில் உள்ள கட்டளை நிரல்கள் எல்லாம் இம்மாதிரி பல பகுதிகளாக (சார்புகளாக) பிரிக்கப்பட்டு எழுதப்படும். அதனால், எவ்வளவு விரைவில் துணைமுறைகள் மற்றும் சார்புகளை வைத்து, கணிப்புகளின் முறைகளை, போலிக்குறியீட்டில் எழுதக் கற்றுக் கொள்கிறீர்களோ அவ்வளவுக்கவ்வளவு நல்லது.

இங்கு, துணைமுறையைப் பயன்படுத்தும் இரண்டு உதாரணங்களைப் பார்ப்போம்.

நூறு எண்களை வரிசைப்படுத்தும் கணக்கை மீண்டும் எடுத்துக் கொள்வோம். இதற்கான போலிக்குறியீட்டையும் கட்டளை நிரலையும் சென்ற பாடத்தில் எழுதினோம். அந்தப் போலிக்குறியீட்டை, திரும்பவும் ஒரு முறை பார்ப்போம்.

1. 100 **எண்களைப் படி. அவை** A (1)...A(100).

இங்கு எப்படி 100 எண்களைப் படிப்பது என்ற விவரம் கொடுக்கப்படவில்லை. அதைத் தனியாக ஒரு துணை முறையாக எழுதலாம்.

2. **I என்பது 1 முதல் 99 வரை செய்**

(A (I) முதல் A (100) வரை உள்ளவற்றில் சிறிய எண்ணைக் கண்டுபிடி. அது A (POS) என்போம்.

A(I) ஐயும் A(POS) ஐயும் இடம் மாற்று.

)

இங்கு, சிறிய எண்ணை கண்டுபிடிப்பதும், இடம் மாற்றுவதும் இரண்டு துணைமுறைகளாகக் கருதப்படலாம்.

3. **வரிசையை வெளியிடு.**

இதையும் ஒரு துணைமுறை எனலாம். இதற்கும் விவரங்கள் இங்கு கொடுக்கப்படவில்லை என்பதைக் கவனி. இப்படி விவரங்களைத் தவிர்த்து, போலிக்குறியீட்டை எழுதுவதுதான் எவ்வளவு எளிது!

இப்போது துணைச் செயல்பாடுகளைப் பயன்படுத்தி, இந்தக் கட்டளை நிரலை எழுதுவோம். இதை நாம் ஏற்கனவே எழுதிய கட்டளை நிரலுடன் ஒப்பிட்டுப்பார்.

```
10 DIM A (100)
20 GOSUB 1000: REM SUBROUTINE TO READ
30 FOR I = 1 TO 99
40      GOSUB 2000: REM SUB TO FIND MIN POSITION
```

```
50      GOSUB 3000: REM SUB TO INTERCHANGE
60      NEXT I
70      GOSUB 4000: REM SUB TO PRINT
80      END
1000    REM SUB ROUTINE TO READ
1010    FOR I  =  1 TO 100
1020              PRINT "GIVE THE", I, "THE NUMBER"
1030              INPUT A (I)
1040    NEXT I
1050    RETURN
1060    REM END OF SUB-READ
2000    REM SUB TO FIND MIN FROM A (I) TO A (100)
2010    MIN    =  A(I): POS=I
2020    FOR J  =  I +1 TO 100
2030              IF A (J) < MIN THEN MIN = A (J): POS=J
2040    NEXT J
2050    RETURN
2060    REM END OF SUB MIN
3000    REM SUB TO INTERCHANGE A (I) AND A (POS)
3010    TEMP  =  A (I) : A (I)  =  A(POS) : A(POS)  =  TEMP
3020    RETURN
3030    REM END OF SUB INTERCHANGE
4000    REM SUB TO OUTPUT
4010    FOR I = 1 TO 100
4020          PRINT A (I)
4030    NEXT  I
4040    RETURN
```

இங்கு சில வரிகள் அதிகம் இருந்தாலும், **முதன்மை நிரல்** (Main Program) சிறியதாகவும், என்ன செய்ய வேண்டும் என்பதை மேலெழுந்த வாரியாகக் குறிப்பதாகவும் இருக்கின்றது. இது நாம் பார்த்த போலிக்குறியீடு போலவே இருப்பது குறிப்பிடத்தக்கது. எந்தத் துணைமுறையைச் செய்ய வேண்டும் என்று பெயரைக் குறிப்பிடாமல் வரி எண்ணை வைத்துக் குறிப்பிடுகிறோம். இது ஒரு குறைதான்.

இருந்தாலும் அந்தத் துணைமுறை எதைக் குறிக்கிறது என்பதை உடனே ஒரு குறிப்பு வாக்கியத்தினால் எழுதி வைக்கிறோம்.

நாம் ஏற்கனவே எழுதிய பத்து போலிக் குறியீடுகளில், ஒரு எண் பகுபடா எண்ணா என்று பார்ப்பது, ஒரு எண்ணின் வலதுபுறக் கடைசி இலக்கத்தைக் கண்டுபிடிப்பது போன்றவற்றைத் துணைமுறைகளாக வைத்து, கட்டளைநிரலை எழுதலாம்.

ஒரு எண்ணைப் படித்து, கீழே உள்ள சார்பின் மதிப்பை வெளியிடு. கட்டளை நிரலில் ON...GOSUB வாக்கியத்தைப் பயன்படுத்து.

$$f(x) = \begin{cases} 0, & \text{if } x = 0 \\ x^2 + 7, & \text{if } x > 0 \\ \sin x + \cos x, & \text{if } x < 0 \end{cases}$$

```
 10 PRINT "GIVE X"
 20 ON SGN (X) +2 GOSUB 100, 200, 300
 30 END
100 REM HERE Y IS NEGATIVE
110 PRINT SIN ( X ) + COS ( X )
120 RETURN : REM  END OF SUB
130 REM
200 REM HERE X IS ZERO
210 PRINT 0: RETURN
220: REM END OF SUB
230 REM
300 REM HERE X IS POSITIVE
310 PRINT X  * X + 7
320 RETURN
```

நினைவில் நிற்க

1. ஒரு குறிப்பிட்ட பணியைச் செய்யும் செயல்பாடுகளை, ஒரு துணைமுறையாக எழுதி வைக்கலாம். துணைமுறைக்குள் செல்ல **GOSUB, ON...GOSUB** என்ற வாக்கியங்களை மட்டும் பயன்படுத்த வேண்டும். வெளியே வர **RETURN** என்பதை மட்டுமே பயன்படுத்த வேண்டும்.

2. துணைமுறைக்குள் செல்ல, அதன் முதல் வரியின் எண்ணைப் பயன்படுத்த வேண்டும்.

3. ஒரு துணைமுறை வேறொரு துணைமுறையைக் கூப்பிடலாம். ஆனால், அவ்வாறு கூப்பிடும்போது ஒரு வளையம் தோன்றக் கூடாது.

4. துணைமுறைகளைப் பயன்படுத்தி, போலிக்குறியீடுகளையும், கட்டளை நிரல்களையும் எழுதுவதுதான், கட்டளை நிரல்களை உருவாக்குவதற்கான சரியான, எளியமுறை.

பயிற்சி வினாக்கள்

1. துணைமுறை என்றால் என்ன? அதன் பயன்பாடு என்ன?

2. துணைமுறையைப் பயன்படுத்துவதில் கடைபிடிக்க வேண்டிய விதிமுறைகள் எவை?

3. துணைமுறையைப் பயன்படுத்துவதனால் ஏற்படும் நன்மைகள் இரண்டினைக் கூறு.

4. போலிக்குறியீட்டில் சுருக்கமாக எழுதியதை, கட்டளை நிரலுக்கு மாற்றும்போது துணைமுறைகள் இயற்கையாக உருவாகின்றன என்பதை ஓர் உதாரணத்துடன் விளக்கு.

5. சென்ற பாடத்தில் வரிசை முறைகளைப் பயன்படுத்தி எழுதிய கட்டளை நிரல்களை (பயிற்சி வினாக்கள் உட்பட) துணைமுறைகளைப் பயன்படுத்தி மாற்றி எழுது.

6. கீழே உள்ளவற்றில் தவறுகள் எவையேனும் இருந்தால் சுட்டிக்காட்டு.

அ.
```
100 GOSUB 200
:
200 REM SUB ROUTINE ENDS AT 300
:
230 GOTO 150
:
300 RETURN
:
350 ON X GOTO 400, 250, 450
```

ஆ.
```
100 REM SUB ENDS AT 200
```

```
200 RETURN
:
300 GOSUB 100
```

இ.
```
50 GOSUB 500
:
500 REM SUB ENDS AT 600
:
600 RETURN TO 60
```

ஈ.
```
100 GOSUB 500
:
500 REM SUB ENDS AT 600
:
600 IF A < 3 THEN 560 ELSE 200
```

உ.
```
100 GOSUB 600
:
600 REM SUB ENDS AT 700
:
650 GOSUB 600
660 GOSUB 900
:
700 RETURN
:
900 GOSUB 600
```

ஊ.
```
100 GOSUB MAX
200 ON A$ GOSUB 1000, 2000
```

5.3 சார்புகள்

கணிப்பதில் சார்புகள் ஒரு முக்கிய பங்காற்றுகின்றன. பொதுவான சார்புகள் பலவற்றின் தொகுப்பு ஒரு நூலகமாக பேசிக் மொழிமாற்றியுடன் வரும் என்று ஏற்கனவே பார்த்திருக்கிறோம். சில சமயங்களில், கணிப்பின் சூழ்நிலைக்குத் தக்கவாறு பல புதிய சார்புகள் தேவைப்படும். அவற்றை, கட்டளை நிரலிலேயே வரையறை செய்து பயன்படுத்திக் கொள்ளலாம். சார்பினை ஒரே வரியில் வரையறை செய்யலாம். அல்லது பல வரிகள் தேவைப்படலாம். சில மொழிமாற்றிகள் முதல் வகையை (ஒரு வரி) மட்டுமே அனுமதிக்கும். சார்புகளின் முக்கியத்துவம் கருதி இங்கு இரண்டு வகைகளையும் பார்ப்போம்.

ஒருவரிச் சார்புகள்

ஒரே வரியில் வரையறை செய்யப்படும் சார்புகள் இப்பெயர் பெறுகின்றன. இவ்வாறு வரையறை செய்யும் இலக்கணத்தை,

DEF FNname (parameter list) = expression

எனலாம். இங்கு name என்பது, ஒரு மாறியின் பெயர். சார்பு ஒரு சரத்தைக் கொடுக்கும் என்றால் $ குறியீடு பெயரின் கடைசியில் வர வேண்டும். சார்பின் பெயரில் FN என்பதும் சேர்ந்து வரும்.உதாரணமாக, FNMAX, FNCODE$ ஒரு சார்பானது பல மாறிகளைச் சார்ந்து வரும். இந்த மாறிகளின் பெயர்கள் அடங்கிய வரிசை parameter list எனப்படும். அவற்றுக்கு இடையில் காற்புள்ளிகள் வரும். உதாரணம்

DEF FNSUM (A, B, C) = A + B + C

Parameters என்பதை dummy arguments, formal parameters எனவும் கூறுவர். இந்த சார்பினை நாம் ஏற்கனவே பார்த்த சார்புகளைப் போலவே, எந்த ஒரு கோவையிலும் பயன்படுத்தலாம். உதாரணமாக,

M = X + FNSUM (R, 3.7, Y+2) என்று எழுதலாம்.

அப்போது A என்பது R இன் மதிப்பையும், B என்பது 3.7 எனவும், C என்பது Y + 2 என்ற கோவையின் மதிப்பையும் பெறும். இந்த மூன்று மதிப்புகளும் கூட்டப்பட்டு FNSUM (R, 3.7, Y + 2) என்பதன் மதிப்பாகக் கிடைக்கும்.

ஒரு சார்பினைக் கூப்பிடும்போது, அதாவது பயன்படுத்தும்போது, அந்தச் சார்பு ஏற்கனவே வரையறை செய்யப்பட்டிருக்க வேண்டும். அப்படி வரையறை செய்யும் போது எத்தனை பெராமீட்டர்கள் கொடுக்கப்பட்டுள்ளனவோ, அத்தனைக்கும், அதே வரிசையில் மதிப்புகளைக் கொடுத்து அந்தச் சார்பினை அழைக்க வேண்டும். இந்த மதிப்புகளைக் கொடுக்கக் கோவைகளைப் பயன்படுத்தலாம்.

சாதாரணமாக, ஒரு சார்பு அதன் பெராமீட்டர்களை மட்டும் சார்ந்திருக்கும். சில சமயங்களில், பெராமீட்டர் வரிசையில் குறிப்பிடப்படாத வேறு சில மாறிகளையும் சார்ந்திருக்கலாம். இந்த மாறிகள் கட்டளை நிரலில் பயன்படுத்தப்படுபவையாக இருக்க வேண்டும். சார்பு பயன்படுத்தப்படும் நேரத்தில், இந்த மாறிகள் பெற்றுள்ள மதிப்பு சார்பைக் கணிக்கப் பயன்படுத்தப்படும். உதாரணமாக,

```
10 DEF FNS (A,B) = A+B+C
20 A =3: B=5 : C = 10
30 M = FNS (A,B): PRINT M,
40 C =20 : PRINT FNS (A,B)
```

என்னும் தொடர் 18 28 என்று வெளியீடு செய்யும். இங்கு A, B என்பவை மாறவில்லை. இருந்தாலும் FNS (A, B), அதாவது FNS (3, 5) என்பது, இரு இடங்களில் இருவேறு விடைகளைக் கொடுக்கிறது. சார்பானது பெராமீட்டராகக் கொடுக்கப் படாத C என்ற மாறியைச் சார்ந்ததாக இருப்பதால் இது சாத்தியமாகிறது. கணதத்துறையில் எழுதப்படும் சார்புகள் பெராமீட்டர்களை மட்டுமே சார்ந்திருக்கும் என்பதைக் கவனிக்கவும். கணதத்தில் $f(x) = 2x^2 + 3x + 5$ என்று எழுதினால், f என்ற சார்பு x ஐ மட்டுமே சார்ந்திருக்கும். $x = 2$ என்றால், இன்று, நேற்று, நாளை என்று எப்போது கேட்டாலும் 19 என்ற விடையைக் கொடுக்கும். ஆனால், பேசிக் சார்புகள், ஒவ்வொரு சமயமும் வெவ்வேறு விடைகளைக் கொடுக்கலாம்.

கணித இயல் சார்புகளுடன் ஒப்பு நோக்கும்போது, இன்னொரு வேற்றுமையும் உண்டு. பேசிக் சார்பினை பெராமீட்டர் எதுவும் இல்லாமலேயே எழுதலாம். DEF FNA = X + Y என்பது சரியான வரையறைதான். இங்கு பெராமீட்டர் எதுவும் இல்லை. ஆனால், பெராமீட்டராக இல்லாத x மற்றும் y என்னும் மாறிகளை இது சார்ந்துள்ளது. கணித இயலில் இவ்வாறு எழுதுவதில்லை.

சார்புகளை வரையறை செய்யும்போது கொடுக்கப்படும் பெராமீட்டர்களுக்கும், அந்தக் கட்டளை நிரலில் பயன்படுத்தப்படும் அதே பெயர்களுடைய மாறிகளுக்கும் எந்த வித சம்பந்தமும் இல்லை. உதாரணமாக,

```
10 DEF FNMUL (A, B, C) = A * B + B * C * A + 2 * A
20 A = 2 : B = 7 : C = 8
30 PRINT NFMUL (1, 1, 2),
40 A = 10 : B = 5 : C = 8
50 PRINT FNMUL (1, 1, 2),
60 PRINT FNMUL (B, C, A)
```

என்பது 7 7 180 என்று வெளியிடும். ஏனென்றால்,

FNMUL (5, 8, 10) என்பதுதான் கடைசியாக வெளியிடப்படுகிறது. சார்பில் இருக்கும் பெராமீட்டர் A க்கும், A = 2 என்பதில் இருக்கும் மாறி A க்கும் எந்த சம்பந்தமும் இல்லை.

சார்பு என்பது ஒரு கோவைக்குச் சமம் என்பதை ஒருவரிச் சார்புகள் தெளிவாகக் காண்பிக்கின்றன. பல இடங்களிலும் கோவைகளாக எழுதாமல் சார்பாக எழுதும் போது, கட்டளை நிரலை எழுதுவதும், புரிந்து கொள்வதும் எளிதாகும்.

பலவரிச் சார்புகள்

ஒரு மதிப்பைக் கணக்கிட, சில சமயங்களில் அதிகக் கணிப்புகளைச் செய்ய வேண்டியிருக்கும். அதற்கு, பல வரிகள் தேவைப்படலாம். அப்போது **பலவரிச் சார்புகளைப்** (Multiline functions) பயன்படுத்தலாம். இதற்கும் பலவரிச்சார்புகளுக்கும் அதிக வேற்றுமையில்லை. இரண்டின் நோக்கமும் ஒன்றே - அதாவது ஒரு மதிப்பைக் கணக்கிடுவது.

பலவரிச் சார்பின் வரையறையின் முதல்வரி

DEF FNname (parameter list)

என்று இருக்கும். அதற்கடுத்து, சார்பின் உடற்பகுதி, அதாவது சார்பின் மதிப்பைக் கணிக்கும் பகுதி இருக்கும். கடைசியாக FNEND என்ற வரி சார்பின் வரையறை முடிந்துவிட்டது என்பதைக் குறிக்கும். சார்பின் உடற்பகுதியில் ஓர் இடத்திலாவது, சார்பின் பெயருக்கு ஒரு மதிப்பை அளிக்க வேண்டும். இந்த மதிப்பைத்தான் சார்பானது தன்னைக் கூப்பிட்ட இடத்திற்கு திரும்பக் கொடுக்கும்.

ஒரு சார்புக்குள் ஏற்கனவே வரையறை செய்யப்பட்ட வேறொரு சார்பைக் கூப்பிடலாம். ஆனால் தன்னையே கூப்பிட்டுக் கொள்ளக் கூடாது. ஏனென்றால் இது ஒரு வளையத்தை உருவாக்கும்.

சார்பின் உடற்பகுதிக்குள் அந்தச் சார்பைக் கூப்பிடுவதன் மூலமே நுழைய வேண்டும். அதே போல் FNEND வழியேதான் கட்டுப்பாடு வெளியே வர வேண்டும். வேறெந்த வகையிலும் உள்ளே செல்வதும் வெளியே வருவதும் தவறு.

இந்த விதிகள் துணைமுறைகளுக்குக் கூறப்பட்டவை போலவே இருப்பதைக் கவனியுங்கள். ஒருவரிச் சார்புகளுக்கு ஏற்கனவே பார்த்த விதிகள் எல்லாம் பலவரிச் சார்புகளுக்கும் பொருந்தும்.

சார்புகளைப் பயன்படுத்தும் சில உதாரணங்களைப் பார்ப்போம்.

1. n, r என்ற இரண்டு எண்களைப் படி. "C" அல்லது "P" என்ற இரண்டில் ஓர் எழுத்தைக் கேட்டுப் பெறு. "C" என்றால் nCr என்பதையும் "P" என்றால் nPr என்பதையும் வெளியீடு செய்.

nCr = n ! / (n - r) ! r !

nPr = n ! / (n - r) !

n! = 1 x 2 x 3 x ... x n

இங்கு n! என்பதை ஒரு சார்பாகவும், nCr, nPr என்பவற்றையும் சார்புகளாகவும் எழுதலாம். கடைசி இரண்டும், முதல் சார்பைக் கூப்பிடும். சார்புகளைப் பயன்படுத்துவதற்கு முன்னால் அவற்றை வரையறை செய்துகொள்ள வேண்டும். அதனால் முதலில் n! என்பதற்கான சார்பை வரையறை செய்ய வேண்டும்.

```
10 DEF FNFACT (N)
20     PROD = 1
30     FOR I = 1 TO N
40         PROD = PROD * I
50     NEXT I
60     FNFACT = PROD
70 FNEND
100 DEF FNCOMB (N,R)
110     FNCOMB = FNFACT (N) / (FNFACT (R) * FNFACT (N - R))
120 FNEND
200 DEF FNPER (N,R)
210     FNPER = FNFACT (N) / FNFACT (N - R)
220 FNEND
300 PRINT "GIVE N AND R"
310 INPUT X,Y
320 PRINT "GIVE P OR C"
330 INPUT OPTION
340 IF OPTION = "P" PRINT FNPER (X, Y)
350 IF OPTION = "C" PRINT FNCOMB (X, Y)
360 END
```

2. NAME$ என்ற வரிசையில் 100 பெயர்களும், AGE என்ற வரிசையில் அவர்களின் வயதும் இருக்கின்றன. இவர்களில் K என்ற எழுத்தில் ஆரம்பிக்கும் பெயர்களைக் கொண்டவர்களாகவும், வயது 20க்கு மேலும் 25க்குக் கீழும் உள்ளவர்களாகவும் இருப்பவர்களை மட்டும் தேர்ந்தெடுத்து அவர்கள் பெயர்களையும், வயதையும் வெளியிடு. தேர்ந்தெடுப்பதை ஒரு சார்பாக எழுது.

ஒரு பெயரையும், வயதையும் வைத்து தேவையா இல்லையா (1 அல்லது 0) என்று கூற ஒரு சார்பு தேவை. அதை முதலில் எழுத வேண்டும்.

```
10 DEF FNNEEDED (X$, M)
20      IF (X$ <"K") OR (X$ > = "L") THEN 150
30      IF (M < = 20) OR (M > = 25) THEN 150
40      FNNEEDED = 1 : GOTO 200
150     FNNEEDED = 0
200 FNEND
210 FOR I = 1 TO 100
220     PRINT "GIVE AGE AND NAME"
230     INPUT AGE (I), NAME$(I)
240 NEXT I
250 REM OUTPUT ONLY ELIGIBLE PERSONS
260 FOR J = 1 TO 100
270     IF FNNEEDED (NAME$ (J), AGE (J)) = 0 THEN 300
280     PRINT NAME$ (J), AGE (J)
300 NEXT J
310 END
```

3. ஒரு தசம இலக்கத்தைப் பெற்று அதை எழுத்தில் மாற்றிக் கொடுக்க ஒரு சார்பு எழுது.

```
10 DEF FNWORD$(D)
20 ON D +1 GOTO 100, 110, 120, 130, 140, 150, 160, 170, 180, 190
100 FNWORD = "ZERO"  : GOTO 200
110 FNWORD = "ONE"   : GOTO 200
120 FNWORD = "TWO "  : GOTO 200
130 FNWORD = "THREE" : GOTO 200
140 FNWORD = "FOUR"  : GOTO 200
150 FNWORD = "FIVE"  : GOTO 200
160 FNWORD = "SIX"   : GOTO 200
170 FNWORD = "SEVEN" : GOTO 200
180 FNWORD = "EIGHT" : GOTO 200
```

```
190 FNWORD  = "NINE"
200 FNEND
300 PRINT "GIVE A DIGIT"  :  INPUT N
310 PRINT N, "IS", NFWORD (N)
320 END
```

இங்கு 5 என்ற இலக்கத்தை உள்ளீடு செய்தால்,

5 IS FIVE

என்று வெளியீடு வரும்.

சார்புகளையும், துணைமுறைகளையும் ஒப்பிட்டு, ஒற்றுமை வேற்றுமைகளைப் பட்டியலிடு - என்பது பல தேர்வுகளில் பல முறைகள் கேட்கப்பட்ட கேள்வி. சில ஒற்றுமை வேற்றுமைகளை இங்கு பார்ப்போம்.

ஒற்றுமைகள்

சார்புகள்	**துணைமுறைகள்**
1. பெயரைக் கூறுவதன் மூலம் மட்டுமே கட்டுப்பாடு இதனுள் செல்ல முடியும்.	GOSUB என்பதன் மூலமே உள்ளே செல்ல முடியும்
2. பலவரிச் சார்பில் FNEND மூலமே வெளிவர வேண்டும்.	RETURN மூலமே வெளிவர வேண்டும்.
3. கூப்பிட்ட இடத்திற்குத் தானாகத் திரும்பும்	கூப்பிட்ட இடத்திற்கு அடுத்துஉள்ள வாக்கியத்திற்குத் தானாகத் திரும்பும்.
4. மற்ற சார்புகளை, துணைமுறை களை, கூப்பிடலாம். ஆனால் வளையம் தோன்றக் கூடாது.	அப்படியே
5. கட்டளை நிரலில் உள்ள மாறி களைப் பயன்படுத்தலாம்.	அப்படியே.
6. இது கோவையின் பொது வடிவம்.	இது கட்டளையின் பொது வடிவம்

வேற்றுமைகள்

சார்புகள்	**துணைமுறைகள்**
1. பெயர் உண்டு.	பெயர் கிடையாது.
2. ஒரு மதிப்பைக் கணக்கிட்டுத் திரும்பக் கொடுக்கும்.	பல மதிப்புகளை மாற்றலாம். ஆனால், எந்த ஒரு மதிப்பையும் தனியாகக் கொடுக்காது.

3. சார்பின் பெயர், அதன் உடலில் ஓர் இடத்திலாவது மதிப்பைப் பெற வேண்டும்.	*பெயர் கிடையாது.*
4. பெராமீட்டர்கள் உண்டு.	*கிடையாது.*
5. ஒரு மதிப்பைக் கணக்கிடும் போது வேறு சில மதிப்புகள் மாற்றப் படலாம். இது **பக்க விளைவு** (Side Effect) *எனப்படும்.*	*மதிப்புகள் மாற்றப்படும்போது அவை எல்லாமே நேர்விளைவுகள்தான்.*

நினைவில் நிற்க

1. சார்புகளை வரையறுப்பதில் இரண்டு வகைகள் உண்டு. ஆனால், பயன்பாடு ஒன்றே.

2. சார்புகள், பெராமீட்டர்கள் எனப்படும் பல மாறிகளின் பெயர்களைச் சார்ந்து இருக்கும். இது ஒரு மதிப்பைக் கணக்கிட்டுக் கொடுக்கும்.

3. சார்புகளுக்கு உள்ளேயும், வெளியேயும் செல்ல, கட்டுப்பாடுகள் உண்டு.

4. சார்புகளுக்கும், துணைமுறைகளுக்கும் பல ஒற்றுமைகள் உண்டு.

5. சார்பின் பெயரில் $ குறி இருப்பதை வைத்து, கிடைக்கும் மதிப்பு சரமா அல்லது எண்ணா என்பதைக் கூறலாம்.

பயிற்சி வினாக்கள்

1. ஒருவரிச் சார்பிற்கும், பலவரிச் சார்பிற்கும் உள்ள இரண்டு ஒற்றுமைகள் இரண்டு வேற்றுமைகளைக் கூறு.
2. சார்பினை வரையறை செய்யும் இலக்கணத்தைக் கூறு.
3. சார்பினைப் பயன்படுத்துவதில் பெராமீட்டர்களுக்கான விதிகளைக் கூறு.
4. சார்பினைப் பயன்படுத்துவதில் உள்ள இரண்டு நன்மைகள் யாவை?
5. சார்பிற்கும், துணைமுறைக்கும் உள்ள ஒற்றுமை, வேற்றுமைகள் யாவை?
6. ஒருவரிச் சார்பிற்கு 5 உதாரணங்கள் கொடு.
7. கீழே காணும் மதிப்புகளைக் கணக்கிட பேசிக் மொழியில் சார்புகள் எழுது. ஒவ்வொன்றையும் எப்படிக் கூப்பிட வேண்டும் என்பதற்கு உதாரணங்கள் கொடு.

அ. $f(x, y) = \begin{cases} x^2 + y^2 & \text{if } x > 0 \text{ and } y > 0 \\ 2x + \cos y & \text{if } x > 0 \text{ and } y <= 0 \\ 4y & \text{if } x <= 0 \end{cases}$

ஆ. ஒரு நுழைவுத் தேர்வில் பதிவு எண் இவ்வாறு கொடுக்கப்படுகிறது. முதல் இலக்கம் எந்தப் படிப்பிற்கான விண்ணப்பம் என்பதைக் குறிக்கும். M(MCA) என்றால் 1, D(DCA) என்றால் 2, B (Both MCA and DCA) என்றால் 3. இரண்டாம் இலக்கம் அவர் வகுப்பைக் குறிக்கும். SC -1, ST - 2, MBC - 3, BC - 4, FC - 5. அடுத்த மூன்று இலக்கங்கள் அவர் அனுப்பும் விண்ணப்பத்தாள் எண். பதிவு எண்ணைக் கணக்கிட ஒரு சார்பு தேவை.

இ. மூன்று எண்களில் சிறியதைக் கண்டுபிடி.

ஈ. நான்கு எண்களின் சராசரியைக் கண்டுபிடி.

உ. ஒரு செவ்வகத்தின் பரப்பளவைக் கண்டுபிடி.

8. கீழே உள்ளவற்றில் தவறுகள் எவையேனும் இருந்தால் சுட்டிக்காட்டு.

அ.
```
10 REM DEF FNA (X,Y)
20 A = X,Y
30 RETURN
40 REM
50 M = FNA (2, 3)
```

ஆ.
```
10 FNAB$ (X$, Y$)
20 FNAB$ = Y$
30 FNEND
```

இ.
```
10 DEF A (X, M)
20 FNA = X * 2 + M
30 FNEND
40 FNA (3, 2) = L
50 PRINT L
60 GOTO 20
70 END
```

ஈ
```
10 DEF FNX$
20 FNX$ = NAME$
30 FNEND
```

```
40 NAME$ = "HIMALAYAS"
50 PRINT FNX$
60 END
```

உ.

```
10 DEF FNAA (X)
20        FNAA (X) = X* 2
30 FNEND
40 Y = FNAA (X,X)
```

ஊ.

```
10 DEF FNR$ (A$, M, B$)
20        FNR$ =  M * Y
30 FNEND
40 FEF FNRR (R,L)
50              RR = R*L*R*L
60 FNEND
70 PRINT FNRR ("X",5)
80 PRINT    (5, "X", "Y")
```

எ.

```
10 DEF FNAB
20              FNAB = Y * M
30              YY = FNAB * Y
40 FNEND
```

5.4 முறையில்லா எண்கள்

கணிப்பொறி நம்முடன் தாயக்கட்டம், வணிகம் (Trade) அல்லது அது போன்ற ஒரு விளையாட்டை விளையாட ஒரு கட்டளை நிரல் எழுத வேண்டும் என்போம். இங்கு, கணிப்பொறி ஒரு பகடைக்காயை உருட்டினால் என்ன எண் வருமென்று கணிக்க வேண்டும். வரும் எண் 1, 2, 3 அல்லது 4 என இருக்க வேண்டும் எனலாம். இவற்றில் எது வேண்டுமானாலும் வரலாம். ஒவ்வொரு முறை ஆடும்போதும் வித விதமாக வர வேண்டும். இவ்வாறு வருவதில் எந்தவித விதி முறையும் இருக்கக் கூடாது. அதாவது விதிமுறைகள் எதற்கும் கட்டுப்படாமல், (குருட்டாம் போக்கில்) 1, 2, 3, 4 என்பவற்றில் ஒன்றைத் தேர்ந்தெடுக்க வேண்டும். இவ்வாறு விதிமுறை ஏதுமின்றி தேர்ந்தெடுக்கப் படும் எண்கள் **முறையில்லா எண்கள்** (Random) எனப்படும்.

இம்மாதிரி விதிமுறை ஏதுமின்றி 0 முதல் 1 வரை உள்ள பின்னங்களில் ஒன்றைத் தேர்ந்தெடுத்துக் கொடுக்க, RND என்ற ஒரு நூலகச் சார்பு உள்ளது. எப்போது ஒரு சார்பைப் பயன்படுத்துகிறோமோ, அப்போது குறிப்பிட்ட ஒரு வழியில்தான் இந்த எண்களை உருவாக்குகிறோம். அதனால் இவை உண்மையில் முறையில்லா எண்கள் இல்லை! ஆனால், கிட்டத்தட்ட அந்த மாதிரி இருப்பதால் இவை **போலி முறையில்லா எண்கள்** (Pseudo random numbers) எனப்படும். இந்த எண்கள் பின்னங்களாக இருந்தாலும், அவற்றை எளிதில் முழு எண்களாக மாற்ற முடியும். 1 முதல் 4 வரை உள்ள முழு எண்களைப் பெற நாமே ஒரு சார்பினை எழுதலாம்.

```
DEF FNRANDOM = INT (RND * 4 )  + 1
```

இப்போது RND சார்பினைப் பயன்படுத்தும் முறையைப் பார்க்கலாம். உதாரணமாக,

```
10 X = RND : Y = RND : Z = RND
20 PRINT X, Y, Z : END
```

என்ற கட்டளை நிரல் 0.1213501, 0.651861, 0.8688611 என்று வெளியிடுவதாக வைத்துக் கொள்வோம். இந்தக் கட்டளை நிரலை எத்தனை ம்றை செயல்படுத்தினாலும், இதே விடைதான் கிடைக்கும். அதாவது RND சார்பை கட்டளை நிரலில் முதல் முறை செயல்படுத்தும்போது ஒரு குறிப்பிட்ட எண்ணைக் கொடுக்கும். இரண்டாம் முறை இன்னொரு குறிப்பிட்ட எண்ணைக் கொடுக்கும். இவ்வாறு, ஒரு வரிசையைத் தனக்குள் வைத்துக் கொண்டு, அதிலிருந்து ஒவ்வொன்றாக எடுத்துக் கொடுப்பது போல் செயல்படும். இதில் நன்மையும் உண்டு. தீமையும் உண்டு.

ஒரு நிரலை எழுதி, தவறுகளைத் திருத்தும் ஆரம்ப காலத்தில், நிரலானது, செயல்படும் ஒவ்வொரு முறையும் ஒரே விடையைக் கணித்துத் தந்தால்தான், தவறுகள்

எங்கிருக்கின்றன என்று கண்டுபிடிப்பது எளிது. ஒவ்வொரு முறையும் வெவ்வேறு விடை கிடைத்தால் தவறுகளைக் கண்டுபிடிப்பது மிகமிகக் கடினமான காரியமாகி விடும். அதனால், RND சார்பு ஒரு வரிசைப்படி எண்களைக் கொடுப்பதில் ஒரு நன்மை இருக்கிறது.

ஆனால், நிரலைச் செம்மைப்படுத்திய பிறகும், இதே போல் வந்தால் என்ன ஆகும்? மேலே சொன்ன விளையாட்டுகளில் கணிப்பொறி ஒவ்வொரு முறையும் என்ன செய்யும் என்று நமக்குத் தெரிந்து விடும், ஒரு ஆட்டம் முடிந்த உடனேயே. அதற்குப்பிறகு நாம் கணிப்பொறியை எளிதில் வென்றுவிடலாம். ஆனால் இது விளையாட்டாக இருக்காது. போலியாக, கேலிக்கூத்தாக மாறிவிடும். என்ன நடக்கப் போகிறது என்பது தெரியாமலிருந்தால்தானே விளையாட்டு - ஏன்? வாழ்க்கையும் கூட - சுவையாக இருக்கும்.

நிரலைச் செயல்படுத்தும் ஒவ்வொரு முறையும் RND சார்பானது புதிய புதிய எண்களுடன் ஆரம்பித்துக் கொடுக்க வேண்டும். இதைச் செயல்படுத்த ஒரு வழி இருக்கிறது. கட்டளை நிரலில் RAMDOMIZE என்று ஒரு கட்டளையை ஆரம்பத்தில் சேர்க்க வேண்டும். இதற்கு, RND சார்பைக் கூப்பிடும்போது ஒரே இடத்தில் ஆரம்பிக்காதே, வெவ்வேறாக ஆரம்பி, என்று பொருள். இனி, ஒவ்வொரு முறை விளையாடும்போதும் கணிப்பொறி விதவிதமாக விளையாடும். சில மொழி மாற்றிகள் RAMDOMIZE வாக்கியத்தைச் செயல்படுத்தும்போது நம்மிடம் ஒரு **மூல எண்ணைக்** (Seed) கேட்டு வாங்கி, அதைப் பயன்படுத்தி ஆரம்ப எண்ணைத் தேர்ந்தெடுக்கும். முறையில்லா எண்கள், பல சிக்கலான கணக்குகளைத் தீர்க்கும் சிறப்பான பல புதிய உயர் வகை வழி முறைகளிலும் பெரிதும் பயனாகின்றன.

நினைவில் நிற்க

1. போலிமுறையில்லா எண்ணை உருவாக்க RND சார்பு பயன்படுகிறது.

2. ஒவ்வொரு முறை கட்டளை நிரலைச் செயல்படுத்தும்போதும், வெவ்வேறு வரிசைகளில் எண்களை உருவாக்க RAMDOMIZE வாக்கியம் பயன்படும்.

பயிற்சி வினாக்கள்

1. முறையில்லா எண்களை உருவாக்க வேண்டிய அவசியம் என்ன?

2. RND சார்பினை விளக்கு.

3. RAMDOMIZE கட்டளையின் செயல்பாட்டை விளக்கு.

4. 15 முதல் 25க்குள் (15, 25 உட்பட) ஒரு எண்ணைத் தேர்ந்தெடுக்க ஒரு சார்பு எழுது.

5. முறையில்லா எண்ணுக்கும், போலி முறையில்லா எண்ணுக்கும் உள்ள வேறுபாடு என்ன?

5.5 சரங்களின் கணிப்பு

ஆரம்ப காலத்தில், கணிப்பொறியில், எண் கணிப்பிற்கே முக்கியத்துவம் அளிக்கப்பட்டது. பிறகுதான் சரங்களைக் கொண்டு செய்யும் கணிப்பின் அவசியம் அறியப்பட்டது. சரக்கணிப்பு பற்றி இந்தப் பாடத்தில் அறிந்து கொள்வோம். சரங்களின் நீளத்தைக் கணக்கிடுவது, சரங்களிலிருந்து குறிப்பிட்ட பகுதிகளைப் பிரித்தெடுப்பது, எண்களையும் சரங்களையும் ஒன்றிலிருந்து மற்றொன்றாக மாற்றுவது போன்றவை இதில் அடங்கும்.

சரம் என்பது குறியீடுகளின் கோவை. நாம் சாதாரணமாக ஆஸ்கி குறியீடுகளைப் பயன்படுத்துகிறோம். இதில் 65 என்பது Aஐயும், 97 என்பது aஐயும், 48 என்பது சுழியை, 43 என்பது + குறியையும் குறிக்கும். இதை ஒரு சார்பாக எழுதலாம். A$ = CHR$ (43) என்று எழுதினால், A$ என்னும் சரமாறி "+" என்ற மதிப்பைப் பெறும். B$ = CHR$ (51) என்றால், B$ இன் மதிப்பு "3" என்ற சரமாகும்.

CHR$ என்ற சார்பு, 0 முதல் 255 வரை உள்ள எண்களை, அதற்கான ஆஸ்கி குறியீடாக மாற்றும். ஆஸ்கி குறியீட்டை, அதற்கான எண்ணாக மாற்றுவதை ASC என்ற சார்பு செய்யும். இந்தச் சார்பிடம் ஒரு சரத்தைக் கொடுத்தால், அந்தச் சரத்தில் உள்ள முதல் குறியீட்டின் ஆஸ்கி எண்ணைக் கொடுக்கும். உதாரணமாக -

ASC("BIG") என்பதும் ASC ("B") என்பதும் 66ஐக் கொடுக்கும்.

A$ என்பது "4AB" என்றால் , ASC(A$) என்பது 52ஐக் கொடுக்கும்.

ASC ("*") என்பது 42 ஐக் கொடுக்கும்.

CHR$ மற்றும் ASC என்ற சார்புகள் ஒன்றுக்கொன்று எதிர்பக்கமாகச் செயல்படுபவை.

CHR$ (ASC ("M")) என்பது "M" ஐக் கொடுக்கும்.

CHR$ (ASC ("MADRAS")) என்பதும் "M" ஐக் கொடுக்கும்.

ASC (CHR$ (78)) என்பது 78ஐக் கொடுக்கும்.

இரண்டு சரங்களை ஒன்றுக்கடுத்து மற்றொன்றை வைத்து, ஒரு புதிய சரத்தை உருவாக்கலாம். இதற்கு, **தொடர்வரிசைப்படுத்துதல்** (Concatenation) என்று பெயர். இது ஒரு **சரச்செயல்பாடு** (String operation). இதுவும் + குறியீடு மூலமே குறிப்பிடப்படும். உதாரணமாக A$ = "COME", B$ = "HERE" என்றால், A$ + B$ என்ற **சரக்கோவை** (String Expression) "COME HERE" என்ற மதிப்பைப் பெறும்.

ADDRESS$ = NAME$ + DOORNO$ + CITY$

என்பது ஒரு சரியான வாக்கியம்.

-6432 என்பது ஒரு தசம எண். "-6432" என்பது ஒரு சரம். இவ்வாறு ஒரு எண்ணை, அதே தசம இலக்கங்கள் கொண்ட சரமாக மாற்றலாம். இதற்கு STR$ என்ற சார்பு பயனாகிறது. STR$(-6432)என்பது "-6432"என்ற சரத்தைக் கொடுக்கும். STR$ (47) என்பது "47" என்ற சரத்தைக் கொடுக்கும். இங்கு + குறிகளுக்காக இடதுபுறம் ஒரு இடம் ஒதுக்கப்பட்டிருக்கும். STR$ என்ற சார்பிற்கு எதிரான செயலை VAL என்ற சார்பு செய்யும். இது சரத்தை ஒரு தசம எண்ணாக மாற்றும். VAL ("67") என்பது 67 என்ற எண்ணையும், VAL ("-437")என்பது -437 என்ற எண்ணையும் கொடுக்கும். VAL ("24") என்பது 24 என்ற எண்ணைக் கொடுக்கும்.

CHR$, ASC, STR$, VAL என்பவை character, ascii, string, value என்ற வார்த்தைகளிலிருந்து பெறப்பட்டவை. சில பெயர்களில் மட்டும் $ குறியீடு சேர்க்கப்படுகிறது. இதை எப்படி நினைவில் வைப்பது? இதற்கு ஏதேனும் காரணம் உண்டா? இருக்கிறது. வரக்கூடிய விடை ஒரு சரமாக இருக்கும்போது மட்டுமே இப்படி $ குறியீடு சேர்க்கப்படும். இனி வரும் பெயர்களிலும் இதைக் கவனிக்கலாம்.

ஒரு சரத்தின் நீளத்தைக் கணக்கிடும் சார்பு LEN. உதாரணம் - LEN ("GOOD BOY") என்பது 8. LEN ("Xb̸b̸b̸X") என்பது 5. "" என்பது வெற்றுச் சரத்தைக் (Null String) குறிக்கும். இதன் நீளம் சுழி. LEN ("") என்பது 0.

சில சமயங்களில், இரண்டு குறியீடுகளுக்கு நடுவில் குறிப்பிட்ட அளவு இடைவெளிகள் விட வேண்டியிருக்கும். இதற்கு அத்தனை இடைவெளிகள் கொண்ட ஒரு சரம் தேவைப்படும். அத்தனை முறை இடைவெளிச் சாவியைத் தட்டி அந்தச் சரத்தை உருவாக்குவதற்குப் பதிலாக, SPACE$ என்ற சார்பைப் பயன்படுத்தலாம். SPACE$ (15) என்பது 15 இடைவெளிகள் கொண்ட ஒரு சரத்தைக் கொடுக்கும். சில தலைப்புகளை அச்சிடும்போது, முதலில் குறிப்பிட்ட அளவு இடைவெளிவிட, இந்தச்சார்பு பெரிதும் பயன்படும்.

ஒரு சரத்திலிருந்து குறிப்பிட்ட பகுதியை எவ்வாறு பிரித்து எடுப்பது என்பதைப் பார்ப்போம். கொடுத்த சரத்தின் இடதுபக்கத்திலிருந்து, அல்லது வலது பக்கத்திலிருந்து, அல்லது நடுவில் குறிப்பிடும் இடத்திலிருந்து - என்று மூன்று விதங்களில், சரத்தின் பகுதியைப் பிரித்தெடுக்கலாம்.

LEFT$ (String, number), RIGHT$ (string, number), MID$ (string, position, number), என்னும் சார்புகள் இதைச் செய்யும். இங்கு string என்பது கொடுக்கப்படும் சரத்தைக் குறிக்கும் சரக்கோவை. (சரக்கோவையின் மதிப்பு ஒரு சரம்.). இது "JOY", A$ + B$, X$ + "JOY" போன்று இருக்கலாம். number என்பது 0 முதல் 255 வரை இருக்கலாம். அத்தனை குறியீடுகள் எடுக்கப்படும். position என்பது எந்த இடத்திலிருந்து ஆரம்பித்து எடுக்க வேண்டும் என்பதைக் குறிக்கிறது. இது1 முதல் 255 வரை இருக்க வேண்டும். LEFT$ மற்றும் RIGHT$ களில் இது தேவையில்லை என்பது தெளிவாகத் தெரிகிறது.

A$ = LEFT$("MATHEMATICS", 4) என்பது, A$ என்ற சரத்திற்கு "MATH" என்ற மதிப்பை அளிக்கிறது.

A$ = LEFT$ ("LIGHT", 8) என்பது, A$ க்கு "LIGHT" என்ற மதிப்பை அளிக்கும். இங்கு கொடுத்த சரத்தில் 8 குறியீடுகள் இல்லை. அதனால், இருக்கும் குறியீடுகளை மட்டுமே கொடுக்கிறது.

RIGHT$ ("BOY", 0) என்பது ஒரு வெற்றுச் சரத்தையும்,

RIGHT$ ("BOY", 2) என்பது "OY" என்ற சரத்தையும்,

RIGHT$ ("BOY", 7) என்பது "BOY" என்ற சரத்தையும் கொடுக்கும்.

MID$ ("MATHEMATICS", 3, 4) என்பது "THEM" என்பதையும்,

MID$ ("MAN", 7, 4) என்பது வெற்றுச் சரத்தையும்,

MID$ ("MAN", 3, 4) என்பது "N" என்பதையும் கொடுக்கும். கடைசி உதாரணத்தில், தேவையான அளவு குறியீடுகள் இல்லை. அதனால், இருக்கும் வரை கொடுக்கும்.

MID$ சார்பில், கடைசியில் வரும் எண்ணைக் கொடுக்க வேண்டிய அவசியம் இல்லை, அப்போது, குறிப்பிட்ட இடத்திலிருந்து வலது கோடி வரை உள்ள எல்லாக் குறியீடுகளையும் கொடுக்கும். MID$ ("TITLE SONG", 7) என்பது "SONG" என்ற சரத்தைக் கொடுக்கும்.

இதுவரை MID$ என்னும் பெயர் ஒரு சார்பினைக் குறிக்கப் பயன்பட்டது. சார்பாக இல்லாமல், இதை இன்னொரு விதத்திலும் பயன்படுத்தலாம். ஒரு சரத்தின் நடுவில் உள்ள சில குறியீடுகளை மட்டும் மாற்றி அமைக்க, இதை மதிப்பளிக்கும் வாக்கியத்தின் = குறிக்கு இடதுபுறம் பயன்படுத்தலாம். (MID$ சார்பாகப் பயன்படும் போது மதிப்பளிக்கும் வாக்கியத்தில் = குறிக்கு வலதுபுறம் தான் வரும்)

MID$ (string1, m,n) = string2 என்று MID$ வாக்கியத்தை எழுதலாம். இங்கு string1 என்பது சரமாறி (சரக்கோவை இல்லை). String2 என்பது ஒரு சரக்கோவை. m, n என்பவை எண்மானக் கோவைகள். இந்த வாக்கியத்தின் பொருள் string1 என்ற சரமாறி குறிப்பிடும் சரத்தில், m என்ற இடத்திலிருந்து ஆரம்பித்து n குறியீடுகளை மாற்றி அமை. அப்படி மாற்றுவதற்கு string 2 என்ற சரக்கோவை கொடுக்கும் சரத்திலிருந்து முதல் n குறியீடுகளை எடுத்துக் கொள்.

இவ்வாறு செய்யும்போது string1 இன் நீளம் மாறாது. string2 இல் n குறியீடுகள் இல்லையென்றால், இருக்கும் வரை (LEN (string2)) மாற்றும். m > LEN (string 1) என்றால் மாற்றம் எதுவும் நிகழாது. n என்ற எண் (எண்மானக் கோவை).கொடுக்கப்படாதபோது, string2 முழுவதும் பயன்படுத்தப்படும். தேவைக்கு அதிகமாக உள்ள குறியீடுகள் பயன்படுத்தப்படமாட்டா.

பழைய மதிப்பு : M$ = "COMPUTER GRAPHICS"

1. செயல்	:	MID$ (M$, 10, 4) = "KITTEN"
புதிய மதிப்பு	:	M$ = "COMPUTER KITTHICS"
2. செயல்	:	MID$ (M$, 7, 3) = "ING"
புதிய மதிப்பு	:	M$ = "COMPUTINGGRAPHICS"
3. செயல்	:	MID$ (M$, 15, 6) = "RICHER"
புதிய மதிப்பு	:	M$ = "COMPUTER GRAPHIC'
4. செயல்	:	MID$ (M$, 15) = "RICH"
புதிய மதிப்பு	:	M$ = "COMPUTER GRAPHRIC"
5. செயல்	:	MID$ (M$, 1, 4) = "XY"
புதிய மதிப்பு	:	M$ = "XYMPUTER GRAPHICS"

MID$ *வாக்கியத்திலும்,* m *என்பது 1 முதல் 255 வரையிலும்,* n *என்பது 0 முதல் 255 வரையிலும் இருக்கவேண்டும்.*

நினைவில் நிற்க.

1. 0 முதல் 255 வரை உள்ள எண்ணை ஆஸ்கி குறியீடாகவும், ஆஸ்கி குறியீடை ஒரு எண்ணாகவும் மாற்ற CHR$, ASC என்ற சார்புகள் பயன்படும்.

2. தசம எண்ணைச் சரமாகவும், சரத்தைத் தசம எண்ணாகவும் மாற்ற STR$, VAL என்ற சார்புகள் பயன்படும்.

3. LEN என்பது சரத்தின் நீளத்தைக் கொடுக்கும்.

4. SPACES (n) என்பது n இடைவெளிகள் உள்ள சரத்தைக் கொடுக்கும்.

5. சரத்தின் ஒரு பகுதியைப் பெற LEFT$, RIGHT$, MID$ என்ற சார்புகளைப் பயன்படுத்தலாம்.

6. MID$ என்பதை மதிப்பளிக்கும் வாக்கியத்தின் இடதுபுறம் பயன்படுத்தி ஒரு சரத்தின் சில குறியீடுகளை மாற்றலாம்.

பயிற்சி வினாக்கள்

1. LEFT$, RIGHT$ என்ற சார்புகளைப் பயன்படுத்தும் முறையை உதாரணங்களுடன் விளக்கு

2. MID$ என்பதைச் சார்பாகப் பயன்படுத்தும்போது, அது எப்படிச் செயல்படும் என்பதை உதாரணங்களுடன் விளக்கு.

3. MID$ என்பதை எவ்வாறு மதிப்பளிக்கும் வாக்கியத்தின் இடதுபுறம் பயன்படுத்துவது என்று விவரி. உதாரணங்கள் கொடு.

4. MID$ என்பதை இருவிதங்களில், சார்பாகவும், வாக்கியமாகவும் பயன்படுத்தலாம். எவ்வாறு?

5. MID$ சார்பைப் பயன்படுத்தும்போது, ஓர் எண்ணைக்குறிப்பிடாமல் விட்டுவிடலாம். அப்போது MID$ எவ்வாறு செயல்படும்?

6. கீழே உள்ள கட்டளைத் தொகுதி என்ன செய்யும்? ஒவவொரு வரியின் செயல்பாட்டையும் விவரி.

```
10A$ = "FINE EXAMPLE"
20 B$ = "HILL TOP"
30 C$ = "FLOWER"
40 D$ = A$ + SPACE$ (2) + C$
50 E$ = MID$ (A$, 6, 25) + "OF"  + C$
60 S$ = C$ + "S"
70 F$ = LEFT$ (B$, 5)  + S$
80 H$ = RIGHT$ (B$, 3)  + " b̸ " + C$
90 MID$ (H$, 1, 4)  =  LEFT$ (B$, 5)
100 MM$  =  MID$ (C$, 3,  3,)  +  MID$ (B$, 5, 3)
```

7. கீழே உள்ள கட்டளைகளில் தவறுகள் எவையேனும் இருந்தால் கண்டுபிடி.

```
TOP$ = "LEFT"+ LEFT$ ("BIG", 1, 2)
A$ = "LEFT$" + "RIGHT$"
MID$ (H$) = LEFT$ ("AB", 2,1)
S$ = "A" +MID ("TOP", 4, 7)
S$ =A + LEFT$ ("GET", 6)
"A$" = B$
A$ = "B$"
A$ = CHR (44) + VAL$ (55) + LEN$ ("GOOD")
X$ = STR$ (5497)
MAX = VAL ("243")  + 76
M$ = M$ + M$
```

8. ஒரு வார்த்தையைப் (சரத்தை) படித்து, அதில் உள்ள எழுத்துகளை ஒரு வரிக்கு ஒரு எழுத்தாக வெளியீடு செய்.

9. ஒரு வார்த்தையைப் படித்து, அதைத் திருப்பி எழுது. உதாரணமாக ANGEL என்பது LEGNA என்று வரவேண்டும். LEN, FOR, MID$ போன்றவற்றைப் பயன்படுத்து.

10. ஒரு வார்த்தையைப் படித்து அது ஒரு பாலின்ட்ரோமா (Palindrome) என்று கண்டுபிடி. MADAM, DADபோன்றவை பாலின்ட்ரோம்கள்.

(குறிப்பு: FOR I = 1 TO (length / 2)

if mid$ (string, I, 1) <> mid$ (string, length-i+1,1) then go out - it is not palindrome).

(இதை சென்ற கணக்கின் விடையிலிருந்தும் செய்யலாம்.)

11. ஒரு வார்த்தையை 20 குறியீடுகளில் நடுவில் இருக்குமாறு செய்

(குறிப்பு:

SPACE$ (10-length/2) + string + SPACE$ (10-length/2) . length. length என்பது இரட்டைப்படை எண் மற்றும் ஒற்றைப்படை எண் என்று இரண்டுக்கும் இந்தக் குறிப்பு பொருந்துமா என்று பார்.)

12. ஒரு வார்த்தையைப் படித்து, அதில் எத்தனை முறை "K" என்ற எழுத்து வந்திருக்கிறது என்று கண்டுபிடிக்க ஒரு கட்டளை நிரல் எழுது.

13. ஒரு வாக்கியத்தைப் படித்து, அதில் வரும் முதல் வார்த்தையை வெளியிடு. முதல் வார்த்தை முடிந்தவுடன் ஒரு இடைவெளி அல்லது காற்புள்ளி இருக்கும்.

14. ஒரு வாக்கியத்தைப் படித்து, அதிலிருந்து இரட்டைப்படை இடங்களில் வரும் எழுத்துக்களை மட்டும் எடுத்து ஒரு வாக்கியத்தையும், ஒற்றைப்படை இடங்களில் வரும் எழுத்துக்களை மட்டும் வைத்து ஒரு வாக்கியத்தையும் உருவாக்கு.

(குறிப்பு : odd$ = " "

for i = 1 to length step 2

odd$ = odd$ + mid$ (string, i,1))

15. ஒரு வாக்கியத்தைப் படித்து, அதைக் கீழ்க்காணுமாறு வெளியிடு.

```
BEAUTIFUL
BEAU IFUL
BEA   FUL
BE     UL
B       L
```

5.6 முறைப்படுத்திய வெளியீடு

வெளியீட்டை முறைப்படுத்துவதற்கு, எளிய வழியான, காற்புள்ளி மற்றும் அரைப்புள்ளிகளைப் பயன்படுத்தும் விதத்தை, சென்ற பகுதியின் ஆரம்பத்தில் பார்த்தோம். அவை ஓரளவே பயன்படும். TAB என்ற சார்பு பற்றியும், PRINT USING என்ற கட்டளை பற்றியும் இங்கு பார்ப்போம்.

TAB

ஒரு பட்டியலைத் தயார் செய்யும்போது தகவல்கள் ஒன்றின் கீழ் ஒன்றாக, குறிப்பிட்ட இடங்களில் ஆரம்பித்து எழுதப்படுவது தேவையாக இருக்கும். பட்டியலில் ஒரு வரியில் வரும் தகவல்களுக்கு முறையே 5, 30, 10, 10, 10 என்ற அளவில் இடங்கள் தேவை என்போம். இந்தத் தகவல்கள் முறையே 1, 6, 36, 46, 56 என்ற இடங்களில் ஆரம்பிக்க வேண்டும். இதைக் காற்புள்ளி மற்றும் அரைப்புள்ளி கொண்டு ஒழுங்குபடுத்துவது என்பது மிகவும் கடினமான செயல். TAB என்ற சார்பு இதை எளிதாக்கும். TAB (n) என்றால், இடம் சுட்டி இந்த வரியில் n என்ற இடத்திற்குப் போக வேண்டும் என்று பொருள். TAB (46) என்றால் இடம் சுட்டி இந்த வரியில் 46 ஆம் இடத்திற்குப் போய் நிற்கும். அடுத்து வருவதை இந்த இடத்திலிருந்து எழுதும்.

ஒரு சிறு பிரச்சனை, ஏற்கனவே இடம்சுட்டி 46 ஆம் இடத்திற்கு அப்பால் சென்றிருந்தால்? அதே வரியில் இடப்புறம் திரும்பி வருமா? இல்லை. வராது, அடுத்த வரிக்குப் போய், அதில் 46 ஆம் இடத்தில் நிற்கும்.

ஓர் உதாரணத்தைப் பார்ப்போம்.

```
10 PRINT  "S. No" TAB (6) "DESCRIPTION" TAB (36) "PRICE"
          TAB (46) "UNITS" TAB (56) "TOTAL"
20 PRINT 1 TAB (6) NAME (3) TAB (36) 44.30 TAB(46)
         2 TAB (56)  2 * 44.30
```

என்று எழுதினால் பின்வருமாறு வெளியீடு வரும்.

```
S.No.   DESCRIPTION    PRICE   UNITS   TOTAL
1.      SHIRTING       44.30   2       88.60
```

இங்கு NAME (3) என்பதை "SHIRTING" என வைத்துக் கொண்டுள்ளோம். TAB சார்பு PRINT வாக்கியத்தின் கடைசித் தகவலாக வந்தால், கணிப்பொறி அடுத்த வரிக்குச் செல்லாமல் அதே வரியில் தொடர்ந்து எழுதும்.

சரங்கள்

PRINT USING என்ற வாக்கியத்தைக் கொண்டு, சரங்களையும் எண்களையும் தேவையான படிவத்தின் வடிவில் வெளியீடு செய்யலாம். இதற்கு print using the given format என்பது பொருள். சரத்தை முறைப்படுத்தி வெளியிடுவது பற்றி முதலில் பார்ப்போம்.

ஒரு சரத்தை மூன்று விதங்களில் வெளியிடலாம். அவை-

1. முதல் எழுத்து மட்டும்,

2. இடமிருந்து n எழுத்துகள்,

3. முழுச் சரமும்.

இதற்கு முறையே " ! ", " \ƀƀ..ƀ \ " (n-2 இடைவெளிகள்), " & "என்ப வற்றை படிவங்களாகப் பயன்படுத்த வேண்டும்.

PRINT USING என்ற கட்டளையை,

PRINT USING format string; list of expressions என்ற வடிவில் எழுத வேண்டும். format string என்பது நாம் மேலே பார்த்த மூன்று சரங்களில் ஒன்று. list of expressions என்பது வெளியிடப்பட வேண்டிய மதிப்புகளைக் கொடுக்கும் கோவைகள். சில உதாரணங்கள்.

```
10  A$ = "ART": B$ = "MUSIC": C$ = "dance drama"
20  PRINT USING "!" ;  A$, B$, C$
30  PRINT USING "!";  A$; B$ ; C$
40  PRINT USING " \ \ " ;  A$; B$ ; C$,
50  PRINT USING " \ ƀ \  " ;  A$; B$ ; C$
60  PRINT USING "  \ ƀƀ ƀ  \  " ; A$; B$  ; C$
70  PRINT USING " &" ;  A$; B$  ; C$
80  PRINT USING "&" ;  A$, B$, C$
```

இதற்கான வெளியீடு இப்படி இருக்கும். சரம் சின்னதாகவும், கொடுக்கப்பட்ட இடம் பெரியதாகவும் இருந்தால், சரத்திற்கு வலதுபக்கம் இடைவெளிகளால் நிரப்பப்படுவதைக் கவனிக்கவும். இங்கு ஒரு வாக்கியத்தில் ஒரு படிவம் தான் இருக்கிறது. எல்லாத் தகவல்களும் இந்தப் படிவத்தின்படியே வெளியிடப்படும்.

```
AMd
AMdARMUdaƀ ƀ ƀƀƀƀƀ ARTMUSdan
ARTƀƀ MUSICdance
ARTMUSICdanceƀ drama
ARTMUSICdancedrama
```

எண்கள்

எண்களை முறைப்படுத்தி வெளியிட உதவும் சில முக்கிய படிவங்களை மட்டும் இங்கு பார்ப்போம். PRINT USING என்பதன் இலக்கணம் ஏற்கனவே பார்த்ததுதான். இங்கு எண்களுக்கான படிவங்கள் மட்டும் மாறும்.

சாதாரணமாக # குறி (hash symbol) ஒரு தசம இலக்கத்தைக் குறிக்கும்.

புள்ளியானது தசமப் புள்ளியின் இடத்தைக் குறிக்கும்.

முதலில் வரும் + குறி, எல்லா எண்களுக்கும் + அல்லது - குறியை இடும்.

** குறியீடு, எண்களுக்கு முன்னால் இடதுபுறத்தை இடைவெளிகளாக விடாமல் * குறிகளைக் கொண்டு நிரப்பும்.

(தசமப் புள்ளிக்கு இடதுபுறம் இடப்படும் காற்புள்ளியானது, தசமப்புள்ளிக்கு இடதுபுறம் 3, 6, 9 இலக்கங்கள் தள்ளி, தேவையானபோது காற்புள்ளிகள் வைக்க வேண்டும் என்பதைக் குறிக்கும். இது ஆயிரம், மில்லியன், பில்லியன் முறைப்படி பிரிக்கும். ஆயிரம், லட்சம், கோடி என்று இல்லை.

இப்போது சில உதாரணங்களைப் பார்ப்போம்.

```
10 A = 1225.437 : B = -4.5:  C = 2:  D  =  4321.56
20 PRINT USING " ####.#"  ; A,  B, C, D
30 PRINT USING " ####.#"b̸ b̸ "  ; A, B, C, D
40 PRINT USING " +####.#  b̸ "; A, B, C, D
50 PRINT USING " **###.# b̸ b̸ " ; A, B, C, D
60 PRINT USING " #####,."  ; A , B, C, D
```

இதற்கான வெளியீடுகள்.

```
1225.44 b̸b̸-4.50b̸b̸b̸ 2.004321.56
1225.4 b̸b̸b̸b̸b̸-4.5b̸b̸b̸b̸b̸2.0b̸b̸4321.6
+ 1225.44b̸b̸b̸b̸-4.50b̸b̸b̸b̸ + 2.00b̸+4321.56
* 1225.4b̸b̸ *** - 4.5b̸b̸ **** 2.0b̸b̸* 321.6
b̸1,225.4b̸b̸b̸b̸-4.5b̸b̸b̸b̸b̸2.0b̸4, 321.6
```

ஒரு படிவத்திலேயே பல தகவல்களுக்கான இடங்களையும் குறிப்பிடலாம். உதாரணமாக

```
10 A = 23.7 : B$ = ""DANCE"
20 PRINT USING "###.#!&" ; A, B$, B$
```

என்றால், b̸ 23.7 DANCE என்று வெளிவரும்.

```
30 PRINT USING "###.#&"; A, B$, A, B$
```

என்றால், b̸23.7DANCE b̸23.7DANCE என்று வெளிவரும்.

இங்கு தகவல்கள் தீரும் வரை, கொடுத்த படிவம் திரும்பத் திரும்ப பயன்படுத்தப்படும்.

படிவத்தைப் பயன்படுத்தி வெளியீட்டை முறைப்படுத்தும்போது, தகவல்களுக்கு நடுவில் காற்புள்ளி, அரைப்புள்ளி என்ற எதைக் கொடுத்தாலும், கணிப்பொறி அதைப் பொருட்படுத்தாது. கொடுத்த படிவத்தின்படி வெளியிடும்.

567 என்ற எண்ணை வெளியிட குறைந்தது மூன்று இடங்கள் தேவை. இதை வெளியிட # # என்று இரு இடங்களை மட்டும் கொடுத்தால் என்ன ஆகும்? கொடுத்த இடம் போதாது என்பதைக் குறிக்க ஒரு % குறியுடன்,எண்ணை சேதப் படுத்தாமல் முழுவதும் கொடுக்கும். தேவையான இடங்களைத் தானே எடுத்துக் கொள்ளும். இந்த உதாரணத்தில் %567 என்று வெளியிடும். 567.9 என்பதை வெளியிட ## என்ற கொடுத்தால் %568 என முழு எண் வெளிவரும்.

நினைவில் நிற்க

1. வெளியீடுகளை முறைப்படுத்த TAB, PRINT USING என்பவற்றைப் பயன்படுத்தலாம்.

2. சரங்களை மூன்று விதங்களிலும், எண்களைப் பலவிதங்களிலும் வெளியீடு செய்யலாம்.

பயிற்சி வினாக்கள்

1. TAB சார்பினை உதாரணத்துடன் விளக்கு.
2. சரத்தை வெளியீடு செய்வதில் PRINT USING எவ்வாறு பயன்படுகிறது?
3. எண்களை வெளியீடு செய்வதில் #, +, */b போன்ற படிவங்கள் எவ்வாறு பயன்படுகின்றன என்பதை உதாரணங்களுடன் விளக்கு.
4. கீழே உள்ள கட்டளைத் தொகுப்புகள் வெளியீடு செய்வதைத் துல்லியமாகக் கூறு.

அ.
```
 10 A  = 4321.756
 20 B  = 45678.222 : F = - 2.7608
 30 C$ = "TAJ MAHAL" ;  D$ = "MADURAI" ;  E$  =  "TINY"
 40 PRINT USING "# # # # ,.# ƀƀ" ; A, B ,F
 50 PRINT USING " ** # # # . #"; A, F
 60 PRINT USING "+ # # # # . # #"; A, A, B
 70 PRINT USING " # . # "; A
 80 PRINT USING "!" ; C$; D$; E$
 90 PRINT USING " &"; C$ ; E$
100 PRINT USING " \ ƀ \";  E$ ; E$
110 PRINT USING " \ ƀƀƀ\ " ; C$ ; D$
120 PRINT USING " & ! \ &  ƀƀ "; C$, D$, E$, C$
```

5. கீழே உள்ள வாக்கியங்களில் தவறுகள் எவையேனும் இருந்தால் கண்டுபிடி.

அ. PRINT USING "!" ; A$,B

ஆ. PRINT USING "# # " ; A$,C,D

இ. PRINT USING " # # # " ; A,B

ஈ. PRINT USING ".# # " ; B

உ. PRINT USING "# # !" ; X, MAX

ஊ. PRINT USING " * #.# " ; x,y

எ. PRINT USING "+ + # # " ; M,N

ஏ. PRINT USING "++++" ; M,N

ஐ. PRINT USING "# # " ;M,M

ஒ. PRINT USING + # # ; A

ஓ. PRINT USING "& & # # " ; A$,B

6

கோப்புகள்

பயன்படுத்தப்படும் தகவல்கள் அதிக அளவில் இருக்கும்போது, அவை கோப்புகளாக மாற்றப்பட்டு வெளி நினைவகங்களில் வைக்கப்படும். இந்தக் கோப்புகளை எவ்வாறு படிப்பது, இவற்றில் எவ்வாறு வெளியீடுகளை எழுதி வைப்பது போன்ற செய்திகளை இந்தப் பகுதியில் பார்ப்போம். கோப்புகளில் வரிசை அணுகுமுறைக் கோப்புகள், நேரடி அணுகுமுறைக் கோப்புகள் என இருவகை உண்டு. இருவிதக் கோப்புகளைப் பயன்படுத்தும் முறைகளையும், அவற்றின் வேறுபாடுகளையும், தேவைகளையும் தெரிந்து கொள்வோம்.

6.1 கோப்பின் வகைகள்

எந்த ஒரு நிறுவனத்திலும், **கோப்புகள்** (files)மூலமாகத்தான் தகவல்கள் கையாளப்படும். கோப்பின் இரு வகைகளைப் பற்றியும், கோப்புகளை எவ்வாறு திறந்து மூட வேண்டும் என்பது பற்றியும் இந்தப் பாடத்தில் படிப்போம். வரிசை அணுகுமுறைக் கோப்புகளைக் கையாளும் விதத்தை அடுத்ததாகப் பார்ப்போம். நேரடி அணுகுமுறைக் கோப்புகளைக் கடைசியாகத் தெரிந்து கொள்வோம்.

கட்டளை நிரல்களைக் கூட, கோப்புகளில்தான் எழுதி வைக்கிறோம். அவற்றுக்குப் பெயரிடும் முறையை ஏற்கனவே பார்த்திருக்கிறோம். (அதிக பட்சம் 8+3 எழுத்துகள்). தகவல் அடங்கிய கோப்புகளுக்கும் இதே முறையில் பெயரிடலாம். பெயர் விரிவில் BAS என்பதற்குப் பதிலாக DAT (data என்பதைக் குறிக்க) அல்லது வேறு தேவையான விரிவைச் சேர்த்துக் கொள்ளலாம்.

தகவல் கோப்புகளில் **வரிசை அணுகுமுறைக் கோப்புகள்** (sequential access files), **நேரடி அணுகுமுறைக் கோப்புகள்** (random access files)என இருவகை உண்டு. காந்த நாடாவில் வரிசை அணுகுமுறைக் கோப்புகளை மட்டுமே வைக்க முடியும். தகடுகளில் இருவிதக் கோப்புகளையும் வைக்கலாம்.

வரிசை அணுகுமுறைக் கோப்புகளில், தகவல்கள் ஒன்றன்பின் ஒன்றாக ஆஸ்கி குறியீடுகளாக வைக்கப்பட்டிருக்கும். தகவல்கள் எண்களாகவும், சரங்களாகவும் இருக்கலாம். சரங்களின் நீளம் வெவ்வேறாக இருக்கலாம். இதனால், தகவல்களை ஒவ்வொன்றாக வரிசையாகத்தான் எழுத வேண்டும். மூன்றாம் தகவலை எழுதி முடித்த பிறகுதான் நான்காம் தகவல்இருக்கும்இடம் தெரிய வரும். இதே காரணத்தினால் மூன்று தகவல்களைப் படித்து முடித்தால்தான் நான்காம் தகவலைப் படிக்க முடியும்.

வரிசைகளில் உள்ள தகவல்களை எவ்வாறு அணுகுகிறோம்? உதாரணமாக MARKS (5) என்றதுமே, அதை நேரடியாக அணுக முடியும். ஏனென்றால் ஒரு வரிசையில் எல்லாத் தகவல்களுமே ஒரே அளவு நீளம் உள்ளவை. ஓர் எண்ணுக்கு 4 பைட்டுகள். MARKS (O) என்பது 16,000 என்ற முகவரியில் தொடங்கினால், MARKS(5) என்பது 16,000 + (5 x 4) = 16,020 என்ற முகவரியில் தொடங்கும். இங்கிருந்து 4 பைட்டுகளைப் படித்தால் MARKS (5) கிடைத்துவிடும்.

நேரடிஅணுகுமுறைக்கோப்புகளிலும் இதே முறைதான் பயன்படுத்தப்படுகிறது. பல தகவல்கள் தொகுக்கப்பட்டு ஒரு **பதிவேடாக** (Record) மாற்றப்படும். பதிவேடுகள் எல்லாம் குறிப்பிட்ட தகவல்களை வரிசையாகக் கொண்டு ஒரே அளவு நீளமுடன் இருக்கும். பதிவேடுகள் ஒவவொன்றாக, வரிசையாக ஒரு கோப்பில் வெளி நினைவகத்தில் வைக்கப்படும். இங்கு குறிப்பிட்ட பதிவேடு எங்கே இருக்கிறது

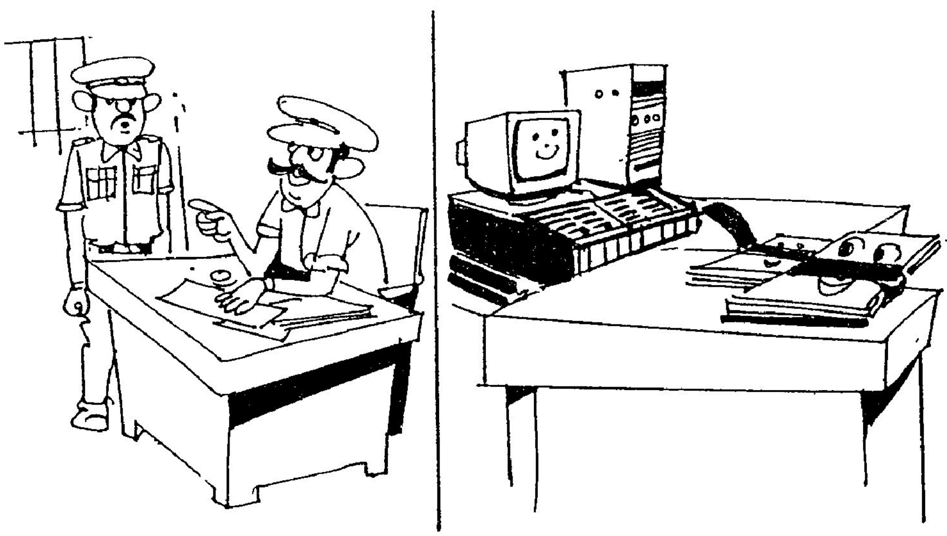
என்பதை மேலே பார்த்ததுபோல் எளிதில் கண்டுபிடிக்கலாம். அதைக் கொண்டு, அந்தப் பதிவேட்டை நேரடியாகப் படிக்கலாம் அல்லது எழுதலாம். இப்படிச் செய்ய காந்த நாடா பயன்படாது. நேரடி அணுகுமுறைச் சாதனமான தகடுகள்தான் பயன்படும்.

ஒரு கட்டளை நிரலில் கோப்பினைப் பயன்படுத்த வேண்டுமென்றால், முதலில், அந்தக் கோப்பு எங்கே இருக்கிறது? அதன் பெயர் என்ன? அதில் எழுதப்போகிறோமா அல்லது படிக்கப் போகிறோமா? அது வரிசை அணுகுமுறைக்கோப்பா, அல்லது நேரடி அணுகுமுறைக் கோப்பா? நேரடி வகை என்றால் பதிவேடுகளின் நீளம் என்ன? - என்னும் எல்லா விவரங்களையும் கூறி, அந்தக் கோப்பினை முறைப்படி திறந்து வைத்துக் கொள்ள வேண்டும். வேலை முடிந்ததும் கோப்பினை மூடி வைக்க வேண்டும். இவை கட்டாயம் பின்பற்றப்படவேண்டிய நியதிகள். இவற்றை எவ்வாறு செய்ய வேண்டும் என்பதையும், அவ்வாறு செய்யும்போது கணிப்பொறி என்ன செய்யும் என்பதையும் ஒவ்வொன்றாகப் பார்ப்போம்.

ஒரு கோப்பினைப் பயன்படுத்தும்போது, ஒவ்வொரு முறையும் அதைப் பெயரிட்டு அழைக்க மாட்டோம். அந்தக் கோப்பிற்கு ஓர் அடையாள எண்ணைக் கொடுத்து, அதையே பிறகு பயன்படுத்துவோம்.

கோப்பினைத் திறக்கும் வாக்கியத்தின் இலக்கணம்--

OPEN file specification FOR mode AS # file number LEN = record length



"301, உள்ளே இருக்கிற ஆளை கூட்டிகிட்டு வாங்க"

"ஏய் கோப்பு எண் 2! உன்னைத்தான். அடுத்த எண்ணையும், அதற்கடுத்த ஒரு சரத்தையும் கொடு"

இங்கு file specification என்பது கோப்பின் முழுப்பெயர். உதாரணமாக, "A: MARKS.DAT", "B:BSC.MKS", "GAME.DAT". இங்கு சிலவற்றில் தகடு இயக்கியின் பெயரும் இடம் பெற்றுள்ளன. கோப்பின் பெயர்கள் சரங்கள் என்பதைக் குறிக்க அவை மேற்கோள் குறிக்குள் வருகின்றன.

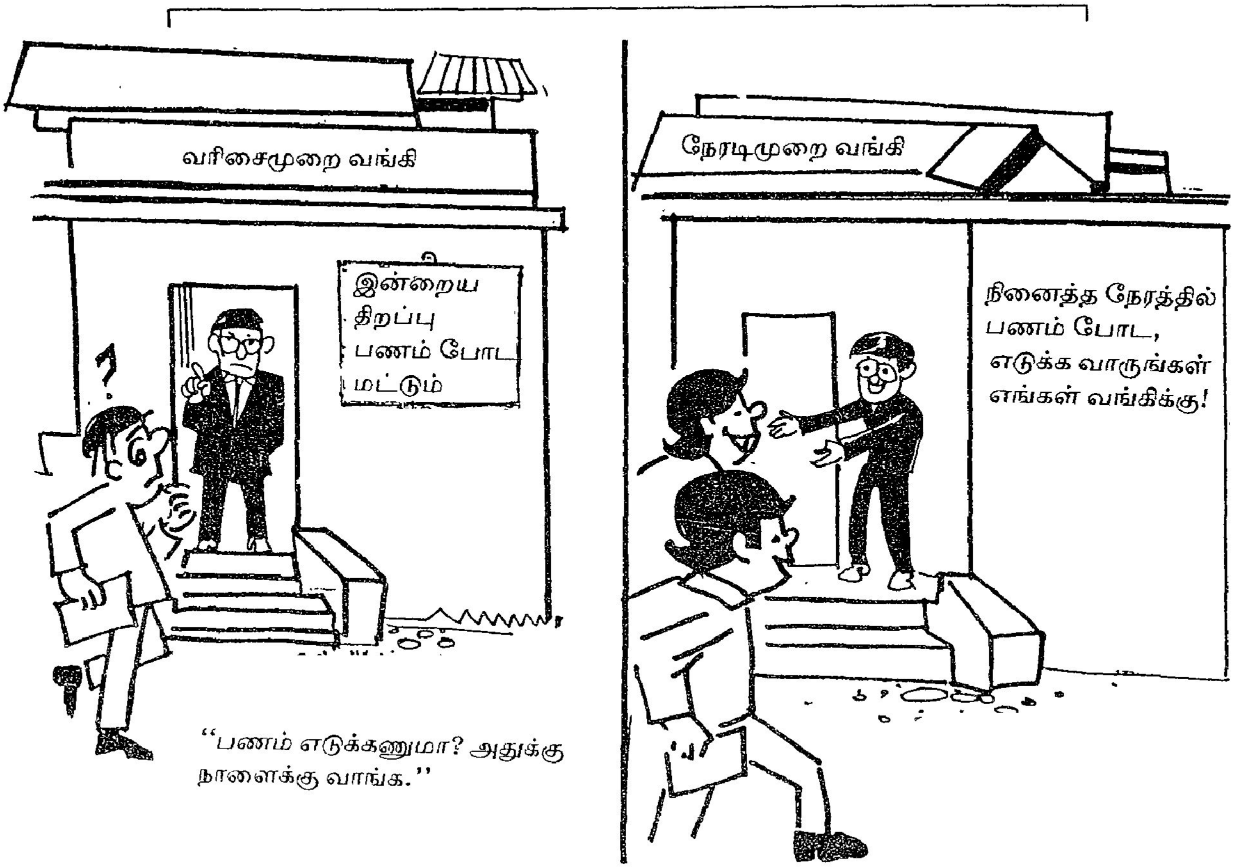
கோப்பினை எதற்காகப் பயன்படுத்தப் போகிறோம் என்பதை mode என்பது குறிக்கிறது. வரிசை அணுகுமுறைக் கோப்பாக இருந்தால் இங்கு INPUT OUTPUTமற்றும் APPEND என்பவற்றுள் ஏதாவது ஒன்றினை மட்டும் குறிப்பிட வேண்டும்.

INPUTஎன்பதற்கு--ஏற்கனவே இருக்கும் இந்தக் கோப்பிலிருந்து வரிசையாகப் படிக்க வேண்டும்; அதனால், கோப்பினைத் திறந்து வைத்து முதல் தகவலைப் படிப்பதற்குத் தயாராக இரு - என்பது பொருள். OUTPUT என்பதற்கு, இந்தக் கோப்பில் வரிசை முறையில் எழுத வேண்டும். அதனால், ஒரு புதிய கோப்பினைத் தயார் செய்து, அதில் முதல் இடத்தில் எழுதத் தயாராக இரு என்று பொருள். ஏற்கனவே அதே பெயருடன் ஒரு கோப்பு இருந்தால் அது அழிக்கப்பட்டுவிடும். அதனால் OUTPUT என்று எழுதும்போது சற்று ஜாக்கிரதையாக இருக்க வேண்டும்.

APPEND என்பதற்கு--இந்தக் கோப்பின் இறுதியில் இன்னும் சில தகவல்களை, வரிசைமுறையில் சேர்க்க வேண்டும்; ஏற்கனவே இருக்கும் இந்தக் கோப்பினைத் திறந்து கடைசி இடத்திற்குப் போய் இரு - என்பது பொருள். இந்தப் பெயரில் கோப்பு எதுவும் இல்லையென்றால், ஒரு புதுக்கோப்பு உருவாக்கப்பட்டு, எழுதுவதற்குத் தயார் செய்யப்படும். காந்த நாடாக்களில் உள்ள கோப்புகளில் இந்த முறையில் கோப்புகளை நீட்ட முடியாது (ஏன்).தகடுகளில் மட்டுமே இவ்வாறு நீட்ட முடியும்.

வரிசை அணுகுமுறைக் கோப்புகளை ஒரு சமயத்தில்எழுதுவதற்கு மட்டும் பயன்படுத்தலாம், அல்லது படிக்க மட்டும் பயன்படுத்தலாம். ஆனால் ஒரே சமயத்தில் இரண்டினையும் செய்ய முடியாது. எழுதத் திறந்த கோப்பினை மூடிவிட்டு, மறுபடியும் படிப்பதற்குக் திறந்தால்தான் அதிலிருந்து படிக்க முடியும்! ஆனால் நேரடி அணுகுமுறைக் கோப்புகளில் ஒரே சமயத்தில் எழுதலாம், படிக்கலாம். இரண்டினையும் மாற்றி மாற்றி நம் விருப்பம் போல் செய்யலாம். அதனால், mode என்பதைக் குறிபிட்ட வேண்டிய அவசியம் இல்லை; FOR என்ற சொல்லையும் விட்டுவிட வேண்டும்.

அடுத்து வரும் file number என்பது சாதாரணமாக 1, 2 அல்லது 3 என இருக்க வேண்டும். ஒரு சமயத்தில் சாதாரணமாக 3 கோப்புகளைத் திறந்து வைத்துப் பயன்படுத்தலாம். அதற்கு மேலும் சில கோப்புகளைத் திறந்து வைக்க சில பேசிக் மொழி மாற்றிகளால் முடியும். ஒரு சமயத்தில் ஒரே எண்ணில் இரண்டு கோப்புகளைத் திறக்கக் கூடாது என்பது பொது அறிவு.



அடுத்து வரும் LEN = record length என்பது நேரடி வகைக் கோப்புகளுக்கு மட்டுமே வரவேண்டும். இங்கு record length என்பது ஒரு பதிவேட்டில் உள்ள மொத்தப் பைட்டுகளின் எண்ணிக்கை. இது *1* முதல் *32,767* வரை இருக்க வேண்டும். நேரடிவகைக் கோப்பினைத் திறக்கும்போது இதைக் கூறாமல் விட்டுவிட்டால், கணிப்பொறி இதை *128* பைட்டுகள் என்று தானாகவே எடுத்துக் கொள்ளும்.

சில உதாரணங்கள்

OPEN "C" : MAX.DAT" FOR OUTPUT AS #2

(வரிசை அணுகுமுறை -- வெளியீடு செய்ய)

OPEN " PAT " AS # 1 LEN = 40

நேரடிமுறை -- பதிவேட்டின் நீளம் 40 பைட்டுகள்)

OPEN " A " AS #3

(நேரடி முறை -பதிவேட்டின் நீளம் 128 பைட்டுகள்)

OPEN "A:BAG " FOR INPUT AS #1

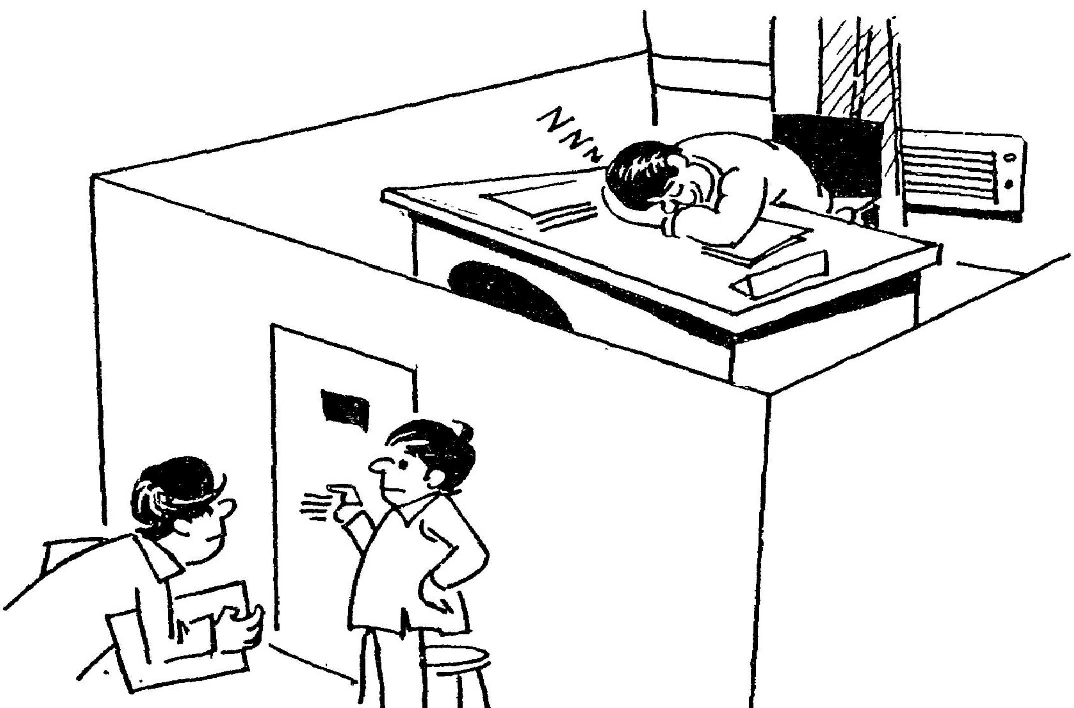
(வரிசை முறை - தகவல்கள் பெற -- ஏற்கனவே உள்ள கோப்பு)

OPEN "XX.MKS " FOR APPEND AS #2

(வரிசை முறை--கோப்பை விரிவுபடுத்த--தகடில் உள்ளது)

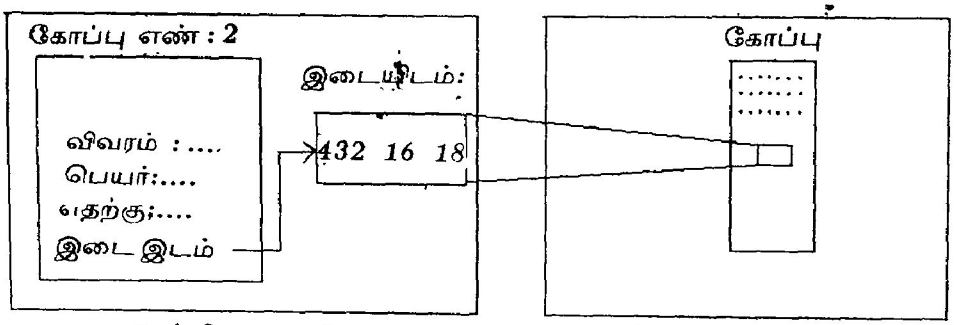
ஒரு கோப்பினைத் திறக்கும்போது என்ன நடக்கிறது என்று பார்ப்போம். கோப்பின் பெயர், அதன் அடையாள எண், எதற்காகத் திறக்கப்பட்டது என்னும் விவரங்கள் ஓர் இடத்தில் எழுதி வைக்கப்படும். இந்தக் கோப்பில் உள்ளீடு அல்லது வெளியீடு செய்ய வசதியாக உள்நினைவகத்தில் ஓர் இடத்தை ஒதுக்கி வைக்கும். இது **இடைஇடம்** (Buffer) எனப்படும். நேரடி அணுகுமுறையில் ஒரு சமயத்தில் ஒரு பதிவேடுதான் எழுதப்படும் அல்லது படிக்கப்படும். அதனால் இடைஇடத்தின் அளவு சாதாரணமாக ஒரு பதிவேட்டின் அளவு இருக்கும். பதிவேட்டின் அளவு குறிப்பிடப்பாடதபோது அது 128 பைட்டுகள் என எடுத்துக் கொள்ளப்படும். வரிசை அணுகுமுறையில் இடைஇடத்தின் அளவு 128 பைட்டுகள்.

இடைஇடம் கோப்பிற்கு ஒரு ஜன்னல் போல் செயல்படும். படிக்கப்படும் தகவல் இந்த ஜன்னல் வழியாகத் தெரிவதாக நினைக்கலாம். எழுதப்படும் தகவல் இந்த ஜன்னல் வழியாகத் தெரியும் இடத்தில் எழுதப்படுவதாக நினைக்கலாம். இந்த ஜன்னல் நிலையானதல்ல! படிக்கும், அல்லது எழுதும் இடத்தைப் பொறுத்து நகர்ந்து கொண்டே செல்லும்! வரிசை முறையில் முதல் தகவலைப் படித்ததும், ஜன்னல் நகர்வதால் அடுத்த தகவல் இந்த ஜன்னலில் தெரியும். இவ்வாறு ஜன்னலை நகர்த்துவது, ஜன்னலில் இருந்து கோப்பில்



"அய்யா ரொம்ப வேலையா இருக்காரு. அவரை நேரடியா பார்க்க முடியாது, உங்களுக்கு எதுவும் வேணுன்னா என்னைக் கேளுங்க. எதாவது கொடுக்கணுன்னா எங்கிட்டே கொடுத்துட்டுப் போங்க!"

எழுதுவது, கோப்பில் இருந்து ஜன்னலில் காண்பிப்பது போன்ற வேலைகளை எல்லாம் கணிப்பொறி தானாகச் செய்துவிடும். நாம் கவலைப்பட வேண்டியதில்லை. கோப்பில் உள்ளீடு வெளியீடு செய்வது எல்லாமே இந்த ஜன்னல் வழியாகத்தான் நடைபெறும்.



படம் 6 - 1 இடை இடம்

ஒரு கோப்பில் தேவையான செயல்பாடுகள் முடிந்துவிட்டால், அதை மூடி விட வேண்டும். அப்போது கோப்பின் விவரங்களும் இடைஇடமும் கணிப்பொறியின் நினைவிலிருந்து நீக்கப்படும். ஒருகோப்புக்கும், அதற்குக் கொடுத்த அடையாள எண்ணுக்கும் இடையே இருந்து இணைப்பை நீக்குவதே இதன் முக்கிய பணி. இதற்குப் பின்னர் ஒரு புதிய OPEN வாக்கியம் மூலம் அந்த எண்ணை வேறு எந்தக் கோப்புடனும் இணைக்கலாம். CLOSE என்னும் கட்டளை, கோப்புகளை மூடும் பணியைச் செய்யும். இதன் பொது வடிவம்.

CLOSE # file number, # file number...

உதாரணங்கள்-

CLOSE #2, #1, #3

CLOSE #3

CLOSE

கடைசிச் சொல்லில் எந்த எண்ணும் கொடுக்கப்படாததால் திறந்திருக்கும் எல்லாக் கோப்புகளுமே மூடப்படும்.

நாம் திறந்த கோப்புகளை நாமே மூடுவது நல்ல வழக்கம். அப்படி மூடாமல் விட்டுவிட்டால், END, NEW, RESET, SYSTEM போன்ற ஆணைகள் இதைச் செய்யும். ஆனால் நாமே மூடுவது சிறந்தது.

நினைவில் நிற்க

1. கோப்புகளில் இருவகை உண்டு, அவற்றைத் திறக்கவும் மூடவும் OPEN, CLOSE என்ற சொற்களைப் பயன்படுத்த வேண்டும்.

2. கோப்பினைத் திறக்க, அதன் பெயர், அடையாள எண், எதற்காகத் திறக்கிறோம், பதிவேட்டின் நீளம் போன்றவை கொடுக்கப்படவேண்டும்.

3. கோப்பில் எழுதுவதும் படிப்பதும் ஒரு இடைஇடம் மூலம்தான் நடைபெறுகின்றன.

பயிற்சி வினாக்கள்

1. வரிசை அணுகுமுறைக் கோப்பிற்கும், நேரடி அணுகுமுறைக் கோப்பிற்கும் உள்ள வேற்றுமைகளைப் பட்டியலிடு.
2. ஒரு வரிசை அணுகுமுறைக் கோப்பினை எவ்வாறு திறப்பது என்று விவரி.
3. ஒரு நேரடி அணுகுமுறைக் கோப்பினைத் திறப்பதற்கான சொல்லினை உதாரணத்துடன் விளக்கு.
4. இடைஇடம் என்றால் என்ன? அதன் அளவு மற்றும் பணிகள் யாவை?
5. கீழ்க்காணும் வாக்கியங்களில் தவறுகள் எவையேனும் இருந்தால் கூறு.

அ. OPEN "R:MAT.DAT" FOR OUTPUT AS #1

ஆ. OPEN "A:MAT.MAT" FOR INPUT AS #2

இ. OPEN X FOR INPUT AND OUTPUT AS #1

ஈ. OPEN "X"FOR #3

உ. OPEN "Y" FOR INPUT AS # 1 LEN =40

ஊ. OPEN "A:A" AS #3 LEN 400

எ. OPEN "M.MAT " AS #2

ஏ. OPEN #2 FOR OUTPUT

ஐ. OPEN X.Y FOR RANDOM AS #77

ஒ. CLOSE ALL

ஓ. CLOSE "A.MAT" AS #2

ஔ. CLOSE #1, #3

6.2 வரிசை அணுகுமுறைக் கோப்புகள்

வரிசை அணுகுமுறைக் கோப்புகளில் எவ்வாறு எழுதுவது, எவ்வாறு படிப்பது என்பவற்றை இந்தப் பாடத்தில் பார்ப்போம். இவ்வகையில் தகவல்கள் ஆஸ்கி குறியீடுகளாக வைக்கப்படும். அவை ஒன்றன் பின் ஒன்றாக வைக்கப்படும். அந்த வரிசையிலேயே படிக்கப்படும். இவற்றை எழுதவும் படிக்கவும் நாம் ஏற்கனவே பார்த்த PRINT, PRINT USING, INPUT வாக்கியங்களை சிறிது மாற்றி பயன்படுத்தலாம். அதற்கு, தகவல்கள் எந்தக் கோப்பில் எழுதப்பட வேண்டும் அல்லது படிக்கப்படவேண்டும் என்பதை மட்டும் இந்த வாக்கியங்களில் கூடுதலாகக் கொடுக்க வேண்டும். இவற்றை,

PRINT # file number, list of expressions

PRINT # file number, USING string; list of expressions

INPUT # file number, list of variables

என்று பொதுவாகக் குறிப்பிடலாம்.

PRINT வாக்கியம், திரையில் எழுதுவதைப் போல அப்படியே கோப்பிலும் எழுதும். அதனால் PRINT வாக்கியத்தில் கொடுக்கும் எண்களுக்கிடையில் காற்புள்ளிக்குப் பதிலாக அரைப்புள்ளி கொடுப்பது நல்லது. இது கோப்பில் இடம் வீணாவதைத் தவிர்க்கும்.

சரங்களை PRINT மூலம் எழுதும்போது சற்று விழிப்புடன் இருக்க வேண்டும். கோப்பிலிருந்து ஒரு சரத்தைப் படிக்கும்போது, கோப்பில் அந்தச் சரம் மேற்கோள் குறிகளுக்கிடையில் இருந்தால் பிரச்சனை எதுவுமில்லை. மேற்கோள் குறிகள் இல்லாவிட்டால், காற்புள்ளியை எல்லைக்குறியீடாக (delimiter) நினைத்து, அடுத்து வரும் காற்புள்ளி வரை படித்து, ஒரு சரமாக எடுத்துக் கொள்ளும். இதனால் வரும் பிரச்சனைகளை ஓர் உதாரணத்தால் விளக்குவோம்.

A$ = "GOOD": B$ = " BOY ": C$ = "ϐ COME, HERE, SOON"

என்று இருக்கட்டும்.

PRINT #1, A$; B$; C$ என்று ஒரு கோப்பில் எழுதிவைப்போம். அந்தக் கோப்பில் இது GOOD BOY COME, HERE, SOON என்று இருக்கும். இதை, INPUT #1, X$, Y$, Z$ என்று படித்தால் என்ன வரும்? X$ = "GOOD BOY COME ", Y$ = "HERE", Z$ = "SOON" என்பது கிடைக்கும். PRINT வாக்கியம் எழுதும்போது மேற்கோள் குறிகள் இல்லாமல் எழுதும். INPUT வாக்கியம் (மேற்கோள் குறிகள் இல்லாததால்) காற்புள்ளியை எல்லைக் குறியாகப் பயன்படுத்தி, தகவல்களைத் தவறாகப் படிக்கும்.

இதைத் தவிர்க்க ஒரு நேரடி வழி, ஒவ்வொரு சரத்தையும் மேற்கோள் குறிகளுக்குள் எழுதுவதுதான். இதற்காக மேற்கோள் குறிகளை நாமே எழுத வேண்டும். இதற்கு, மேலே உள்ள PRINT வாக்கியத்தை இவ்வாறு மாற்றலாம்.

```
PRINT #1, CHR$ (34);  A$; CHR$ (34);  CHR$ (34); B$
          CHR$ (34); CHR$ (34); C$; CHR$ (34)
```

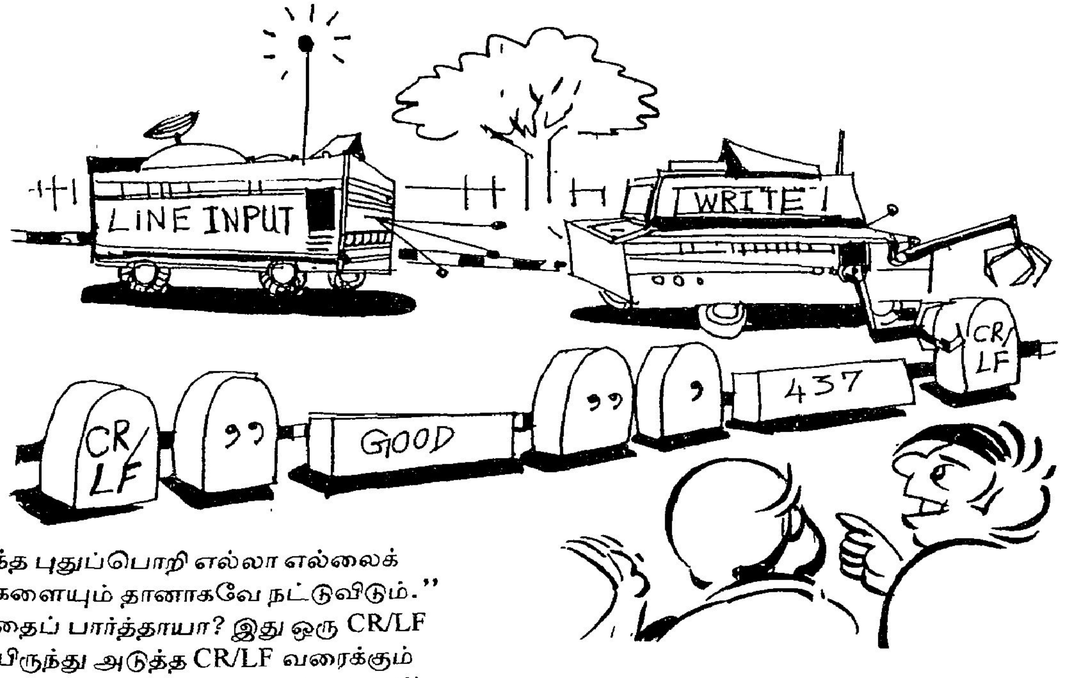
CHR$ (34) என்பது மேற்கோள் குறி. இப்போது கோப்பில் "GOOD" "BOY" "COME, HERE, SOON" என்று இருக்கும். இதைப் படிக்கும்போது தகவல்கள் சரியானபடி கிடைக்கும்.

நாமே இந்த மேற்கோள் குறிகளைப் பல முறை இடுவது பெரிதும் தொல்லையாக இருக்கும். இவற்றைக் கணிப்பொறியே போட்டுவிட்டால் நம் வேலை குறையும். இதற்கென புது வாக்கியம் ஒன்று உண்டு. PRINT என்பதற்குப் பதிலாக WRITE என்று எழுத வேண்டும். அவ்வளவுதான். இது தானாக நான்கு காரியங்களைச் செய்யும்.

1. ஒவ்வொரு சரத்தையும் மேற்கோள் குறிகளுக்குள் எழுதும்.

2. இரண்டு தகவல்களுக்கு இடையில் ஒரு காற்புள்ளி இடும்.

3. நேர்மறை எண்களை இடதுபுறம் இடம் விடாமல் எழுதும்.



"இந்த புதுப்பொறி எல்லா எல்லைக் குறிகளையும் தானாகவே நட்டுவிடும்."
"இதைப் பார்த்தாயா? இது ஒரு CR/LF லேயிருந்து அடுத்த CR/LF வரைக்கும் ஒரே மூச்சில், ஒரே சரமா படிச்சுடும்?"

4. ஒரு வரியை எழுதி முடித்ததும் carriage return/line feed என்ற குறியீடுகளை கூடச் சேர்க்கும்.

உதாரணமாக,

A = 34, B = 65, C$ = "GANDHI, NEHRU" என்றால், WRITE #1, A, B, C$ என்பது,

34, 65, "GANDHI, NEHRU" என்று எழுதும். இந்த WRITE வாக்கியத்தைத் திரையில் எழுதவும் பயன்படுத்திக் கொள்ளலாம். இதற்கு # file number என்பதை மட்டும் நீக்கிவிட வேண்டும்.

வரிசை அணுகுமுறைக் கோப்பிற்கு இன்னும் ஒரு வாக்கியத்தைப் பார்ப்போம். ஒரு கோப்பில் பல முகவரிகள் இருப்பதாக வைத்துக் கொள்வோம். உதாரணமாக, அவை, கா.ஆறுமுகம், 25, முதல் தெரு, திருவல்லிக்கேணி, சென்னை-5, தி.லதா, திருமுல்லைவாயில், தமிழ்நாடு என்பவை போல் இருக்கலாம். ஒவ்வொரு முகவரியிலும் உள்ள குறியீடுகளின் எண்ணிக்கை மாறுபடுகிறது. காற்புள்ளியை எல்லைக் குறியாகப் பயன்படுத்தலாம் என்றால், முதல் முகவரியில் 5 தகவல்களும், இரண்டாவதில் மூன்று தகவல்களும் இருக்கின்றன. அதனால், காற்புள்ளியைப் பயன்படுத்த முடியாது. இதை மொத்தமாகப் படித்து, அதிலிருந்து நாமாக தகவல்களைப் பிரித்து எடுத்துக் கொள்வதுதான் சிறந்த வழி. ஒவ்வொரு **வரியின் முடிவிலும்** Carriage return/line feed இருக்கும். இதை எல்லைக் குறியாகப் பயன்படுத்தி வரிவரியாகப் படிப்பதற்கு

LINE INPUT என்னும் வாக்கியம் பயன்படும்.

LINE INPUT # file number, string variable

என்பது இதன் இலக்கணம். இதைக் கொண்டு, ஒரே ஒரு சரத்தை மட்டும் பெற முடியும். பல சரங்களையோ, எண்களையோ பெற முடியாது.

LINE INPUT #2, A$

LINE INPUT #3, ADDRESS$ (I)

என்பவை சரியான வாக்கியங்கள்.

ஒரு வரியை முழுவதுமாக சாவிப்பலகையில் இருந்து படிப்பதற்கும் LINE INPUT வாக்கியத்தைப் பயன்படுத்தலாம். இதற்கு # file number என்பதை விட்டுவிட வேண்டும். தேவையானால், நினைவூட்டும் சரத்தை சேர்த்துக் கொள்ளலாம். INPUT மற்றும் LINE INPUT வாக்கியங்கள், கணிப்பொறியானது சாவிப்பலகை மூலம் உள்ளீடு கேட்கும்போது மட்டும்தான் திரையில் கேள்விக் குறியைக் காண்பிக்கும். கோப்பிலிருந்து படிக்கும்போது இவ்வாறு காண்பிக்காது.

LINE INPUT "GIVE A$" ; A$ என்பது சரியான வாக்கியம். A$க்கு உள்ளீடு கேட்கும்போது GIVE A$ என்று திரையில் காட்டி நமக்கு நினைவூட்டும்.

நினைவில் நிற்க

1. வரிசை அணுகுமுறைக் கோப்புகளில், இடம் வீணாவதைத் தடுக்க எண்களுக் கிடையில் அரைப்புள்ளியைக் கொண்டு எழுதலாம்.

2. காற்புள்ளி, மேற்கோள் குறியீடு மற்றும் CR/LF என்பவை எல்லைக் குறிகளாகப் பயன்படும்.

3. சரங்களைப் படிக்கும்போது ஏற்படும் பிரச்சனைகளைத் தீர்க்க WRITE வாக்கியத்தைப் பயன்படுத்துவது நல்லது. இந்த வாக்கியம் 4 செயல்களைத் தானாகச் செய்யும்.

4.LINE INPUT என்பதுகோப்பிலிருந்து அல்லது சாவிப்பலகையிலிருந்து வரி வரியாகப் படிக்கும்.

5. WRITE வாக்கியத்தைத் திரையில் எழுதவும் பயன்படுத்தலாம்.

பயிற்சி வினாக்கள்

1. PRINT வாக்கியத்தினால் கோப்பில் எழுதும் முறையை விவரி

2. WRITE வாக்கியத்தின் முக்கியத்துவம் என்ன? அதன் செயல்பாட்டை விவரி.

3.வரிசை அணுகுமுறையில் ஒரு கோப்பிலிருந்து சரங்களைப் படிக்கும்போது ஏற்படக்கூடிய சிக்கல்கள் இரண்டினைக் கூறு.

4. மேற்கோள் குறிகளை ஏன் எல்லைக் குறிகளாகப் பயன்படுத்த வேண்டும்?

5. எல்லைக் குறி என்றால் என்ன? அதனால் என்ன பயன்?

6. கீழ்க்காணும் வாக்கியங்களில் தவறுகள் எவையேனும் இருந்தால் கண்டுபிடி.

அ. PRINT A, B, C, #1

ஆ. PRINT #1, A#2, B

இ. PRINT#1:A$, B, C$

ஈ. INPUT#B, 123

உ. LINE INPUT#1; A$, B$, C$

ஊ. LINE INPUT #1, A

எ. WRITE USING "##", #2, B

ஏ. WRITE#1,USING "##"A, B

ஐ. WRITE#3, A, B$

ஒ. WRITE#3, A, B

ஓ. PRINT #2,USING "##&##"; A, B$, C

6.3 வரிசை முறை உதாரணங்கள்

வரிசை அணுகுமுறைக் கோப்புகளை உருவாக்குவதையும், பயன்படுத்துவதையும் தெளிவாகத் தெரிந்துகொள்ள, பல உதாரணங்களை இங்கு பார்ப்போம். முதலில் ஒரு கோப்பினை உருவாக்குவோம்.

1. ஒரு வகுப்பின் விவரங்களை ஒரு கோப்பில் எழுதி வைக்க வேண்டும். இந்தக் கோப்பில், முதலில், வகுப்பின் பெயர் (உதாரணம், "PLUS TWO", "TENTH"), வகுப்பாசிரியர் பெயர், மாணவர்கள் எண்ணிக்கை போன்ற விவரங்கள் இருக்க வேண்டும். அதற்குப் பிறகு, ஒவவொரு மாணவனுக்கும், பெயர், வகுப்பு எண், ஐந்து பாடங்களில் பெற்ற மதிப்பெண்கள் இருக்க வேண்டும். இந்த விவரங்களை சாவிப்பலகை வழியாக உள்ளீடு செய்ய வேண்டும்.

முதலில் போலிக் குறியீட்டில் எழுதுவோம்.

1. கோப்பினை வரிசை அணுகுமுறையில், எழுதுவதற்குத் திற.
2. வகுப்பின் பெயர் போன்றவற்றைத் திரையில் பெறு.
3. இவற்றைக் கோப்பில் எழுது.
4. 1. முதல் மாணவர் எண்ணிக்கை வரை செய்

   விவரம் கேட்டு, கோப்பில் எழுது
5. கோப்பினை மூடு

```
10 OPEN "A : MARKS. DAT" FOR OUTPUT AS # 1
20 REM
30 INPUT "GIVE CLASS NAME " : CLASS$
40 INPUT "TEACHER'S NAME " ·T$
50 INPUT "NUMBER OF STUDENTS" : STD
60 WRITE#1, CLASS$ ; T$ ; STD
70 REM
100 FOR 1=1 TO STD
110 INPUT "NAME" ; NAME$
120 INPUT "ROLL NO"; RNUMBER
130 INPUT "FIVE MARKS", M(1), M(2), m(3),m(4),m(5)
140 WRITE#1, NAME$ ; RNUMBER;M(1);M(2);M(3);M(4);M(5)
150 NEXT I
160 REM
200 CLOSE #1
210 END
```

பிரச்சனைகள் வராமலிருக்க PRINTஎன்பதற்குப் பதிலாக WRITE பயன்படுத்தப் பட்டிருக்கிறது. இடத்தை வீணாக்காமல் இருக்க, WRITE வாக்கியத்தில் எண்களுக் கிடையில் அரைப்புள்ளி பயன்படுத்தப்பட்டிருக்கிறது.

2. மேலே தயாரித்த கோப்பினைப் படித்து மாணவர் விவரங்களைப் பட்டியலாக வெளியிடு.

1. கோப்பினைப் படிப்பதற்கு, வரிசைமுறையில் திற.

2. வகுப்பின் பெயர், ஆசிரியர் பெயர், மாணவர் எண்ணிக்கை என்பனவற்றைக் கோப்பிலிருந்து படி.

3. வகுப்பின் பெயர், ஆசிரியர் பெயர் இவற்றைத் திரையில் எழுது. தலைப்பை எழுது.

4. 1 முதல் மாணவர் எண்ணிக்கை வரை செய்.

கோப்பிலிருந்து மாணவர் விவரம் படித்து, அவற்றைத் திரையில் எழுது.

5. கோப்பினை மூடு.

```
10 OPEN "A: MARKS.DAT" FOR INPUT AS #1
20 REM
30 INPUT #1, CLASS$, T$, NUM
40 PRINT "CLASS:"; CLASS$
50 PRINT "TEACHER:"; T$
60 PRINT TAB (20) "STUDENTS MARK LIST"
70 PRINT "S.NO, NAME"; TAB (41)"ROLL NO",
80 PRINT "SUB1 SUB2 SUB3 SUB4 SUB5"
90 REM
100 FOR I = 1 TO NUM
110     INPUT #1, N$, RNO
120     PRINT USING "# # # # # &"; I; N$
130     PRINT TAB(41); RNO; TAB (51);
140     FOR J = 1 TO 5
150             INPUT #1, M
160             PRINT USING " # # # # # "; M;
170     NEXT J
180     PRINT
190 NEXT I
```

```
200 PRINT "..... "
210 CLOSE
220 END
```

இங்கு பட்டியல் தயார் செய்யும்போது S.No, Name, Roll No என்பவற்றிற்கு முறையே 5, 35, 10 இடங்களும், ஒவ்வொரு மதிப்பெண்ணுக்கும் 5 இடங்களும் விடப்படுகின்றன. ஒரு மாணவனின் விவரங்கள் எல்லாம் ஒரே வரியில் வர வேண்டும் என்பதனால், வரிக்கு 120, 130, 160இல் PRINT வரியின் கடைசியில் அரைப்புள்ளி இருக்கிறது. ஒரு மாணவனுக்கு இவை எல்லாம் அச்சிட்டதும், அடுத்த வரிக்குப் போக வேண்டும்.

வரி 180 இதைச் செய்வதற்காக உள்ளது. அது முதல் வளையத்திற்கு உள்ளேயும், இரண்டாவது வளையத்திற்கு வெளியேயும் இருப்பதைப் பார்.

3. பல முகவரிகள் அடங்கிய நோட்டுப் புத்தகத்தைப் பார்த்து ஒரு கோப்பினைத் தயார் செய். எத்தனை முகவரிகள் இருக்கின்றன என்று எண்ணாமல் இதைச் செய்து முடி.

எண்ணாமல் செய்ய வேண்டுமானால், FORக்குப் பதிலாக WHILE பயன்படுத்தப்பட வேண்டும். ஒவ்வொரு முறையும், மேலும் தொடர வேண்டுமா என்று கேட்டு, தேவையென்றால் செய்ய வேண்டும்.

1. வரிசை முறையில் எழுத கோப்பைத் திற.

2. தொடர வேண்டுமா என்று கேட்டுப் பெறு.

3. (தொடர வேண்டும் = "ஆம்") என்றிருக்கும் வரை செய்

    ஒரு முகவரியை (ஒரு வரியை) கேட்டுப்படி.

    அதைக் கோப்பில் எழுது.

    தொடர வேண்டுமா என்று கேட்டுப்பெறு.

4. கோப்பினை மூடு.

```
10 OPEN "B: ADDR.DAT" FOR OUTPUT AS #1
20 REM
30 INPUT "CONTINUE? SAY Y OR N"; CONT$
40 WHILE CONT$ = "Y"
50       INPUT "GIVE ADDRESS IN ONE LINE"; A$
60       WRITE #1, A$
70       INPUT "CONTINUE? SAY Y OR N"; CONT$
80 WEND
90 REM
```

```
100 CLOSE
110 END
```

4. மேலே உருவாக்கிய கோப்பில் எத்தனை முகவரிகள் இருக்கின்றன என்பதனைக் கண்டுபிடி.

நினைவிருக்கிறதா? DATA வாக்கியத்தில் கொடுத்திருக்கும் தகவல்களை விட அதிக எண்ணிக்கையில் தகவல்களைப் படிக்க முயற்சி செய்தால் OUT OF DATA என்னும் பிழைச் செய்தி வரும். அதே போல், வரிசை அணுகுமுறைக் கோப்பினையும், தகவல்கள் தீர்ந்து போன பிறகு படிக்க முயற்சித்தால், INPUT PAST END என்ற பிழைச் செய்தி வரும். இந்தத் தவறு நேராமல் எப்படிக் காப்பது? கோப்பு தீர்ந்து விட்டதா என்று ஒவ்வொரு முறையும் பார்த்து, தீர்ந்து போகவில்லை என்றால் மேலும் படிக்கலாம். தீர்ந்து விட்டதென்றால் மேலே படிக்காமல் கோப்பினை மூடவேண்டும். இதற்கு உதவுவது EOF (End of file) என்ற சார்பு, EOF(கோப்பு எண்) என்பது இதன் இலக்கணம். கோப்பு தீர்ந்து போனதென்றால் *1* (ஆம்) என்ற மதிப்பையும், இல்லையென்றால் *0* (இல்லை) என்ற மதிப்பையும் கொடுக்கும். இந்தச் சார்பு, வரிசை அணுகுமுறைக் கோப்புகளுக்கு மட்டும்தான் பயன்படும். நேரடி அணுகுமுறைக் கோப்புகளுக்குப் பயன்படாது.

1. வரிசை முறையில் படிப்பதற்கு கோப்பினைத் திற.

2. n = 0

3. NOT EOF என்று இருக்கும் வரை செய்

  கோப்பிலிந்து ஒரு வரியைப் படி

  n = n + 1

4. கோப்பினை மூடு

5. n ஐ வெளியிடு.

இதற்கான கட்டளை நிரல்

```
10 OPEN "ADDRESS.DAT" FOR INPUT AS # 2
20 N = 0
30 REM
40 WHILE (NOT EOF(2))
50       LINE INPUT #2, A$
60       N = N+1
70 WEND
80 PRINT " NUMBER OF ADDRESSES = " ; N
90 CLOSE #2
100 END
```

ஒரு கட்டளை நிரலில் ஒரு கோப்பினைத் தயார் செய்தால், பிறகு அந்தக் கோப்பிலிருந்து அதே நிரலில் படிக்கவும் முடியும். இவ்வாறு செய்வதற்கு, கோப்பில் எழுதி முடிந்தவுடன் அதை மூடிவிட்டு, பிறகு படிப்பதற்காகத் திரும்பவும் அந்தக் கோப்பினைத் திறக்க வேண்டும்.

5. ஒரு நிறுவனத்தில் பணி புரிபவர்களின் பெயர்களையும், அவர்கள் சம்பளத்தையும் கொண்ட ஒரு வரிசை அணுகுமுறைக் கோப்பினைத் தயார் செய். பிறகு, N என்ற எண்ணைப் படித்து, N ரூபாய்களுக்கு மேலாகச் சம்பளம் வாங்குவோர்களின் பெயர்களையும், அவர்கள் சம்பளத்தையும் பட்டியலிடு. இப்படிப்பட்டவர்கள் எத்தனை பேர், அவர்கள் மொத்தச் சம்பளம் எவ்வளவு என்பனவற்றை அந்தக் கோப்பினைப் படித்துக் கண்டுபிடி.

கோப்பினைத் தயார் செய்த பிறகே N என்ற எண் கொடுக்கப்படும். அதனால், கோப்பினை மறுபடி திறந்து, முழுவதும் பார்த்தால்தான் விடை கிடைக்கும்.

1. வரிசை முறையில் எழுத கோப்பினைத் திற.

2. மீண்டும் = ஆம்

3. (மீண்டும் = ஆம்) என்றிருக்கும்வரை செய்.

   பெயர், சம்பளத்தைப் படித்து, அவற்றைக் கோப்பில் எழுது. மீண்டும் செய்ய வேண்டுமா என்றுகேட்டு வாங்கு.

4. கோப்பினை மூடு.

5. கோப்பினை படிப்பதற்குத் திற.

6. Nஐ கேட்டுப் பெறு.

7. மொத்தம் = 0; ஆட்கள் = 0

8. (NOT EOF) என்றிருக்கும் வரைசெய்

   பெயர், சம்பளத்தைப் படி.

   சம்பளம் > N என்றால் திரையில் காண்பி

   மொத்தம் = மொத்தம் + சம்பளம்; ஆட்கள் = ஆட்கள் +1.

9. ஆட்கள், மொத்தத்தை எழுது.

10. கோப்பினை மூடு.

```
10 OPEN "SALARY.DAT" FOR OUTPUT AS #3
20 AGAIN$ = "Y"
30 WHILE (AGAIN$ = "Y")
40        INPUT "GIVE NAME AND SALARY " ; N$, PAY
50        WRITE #3, N$ ; PAY
```

```
 60       INPUT "MORE?  SAY Y or N" ; AGAIN$
 70 WEND
 80 CLOSE # 3
 90 OPEN "SALARY.DAT" FOR INPUT AS #3
100 REM
110 INPUT "GIVE AMOUNT" ; N
120 TOTAL = 0 ; PERSONS = 0 ;
130. PRINT "NAME" TAB (41 ) "SALARY"
140 WHILE (NOT EOF(3))
150       INPUT #3, X$, M
160       IF M > N THEN 170 ELSE 200
170       PRINT X$; TAB(41)
180       PRINT USING "# # # # #. # #";M
190       TOTAL  = TOTAL  + M ; PERSONS = PERSONS +1
200 WEND
210 PRINT "PERSONS = " ; PERSONS, "TOTAL AMOUNT =  " ;TOTAL
220 CLOSE
230 END
```

சம்பளத்தில் வரும் தசமப் புள்ளிகள் ஒன்றின் கீழ் ஒன்றாகப் பொருந்தி வருவதற்காக PRINT USING பயன்படுத்தப்பட்டுள்ளது. 170 ஆம் வரி PRINTவாக்கியத்தில் கடைசியில் TAB இருப்பதால், அடுத்த PRINTஇல் வரும் சம்பளம்,பெயர் எழுதப்பட்ட வரியிலேயே எழுதப்படும். பெயர்கள் பலவித நீளங்களில் இருந்தாலும், அது சம்பளம் எழுதும் இடத்தை பாதிக்கக்கூடாது என்பதற்காகவும் இந்த TAB (41) கொடுக்கப்பட்டுள்ளது.

மேலே ஏற்படுத்திய கோப்பில் மேலும் சில பெயர்களையும், அவர்களின் சம்பளத்தையும் சேர்க்க வேண்டாமென்றால் என்ன செய்ய வேண்டும்? மேலே எழுதிய கட்டளை நிரலின் முதற்பகுதியை ஒரே ஒரு மாற்றத்துடன் அப்படியே பயன்படுத்தலாம். அந்த மாற்றம்,கோப்பினைத் திறக்கும் விதத்தில்தான், தேவையான புதிய வாக்கியம் -

```
10 OPEN "SALARY.DAT" FOR APPEND AS #3
```

நினைவில் நிற்க

1. வரிசை அணுகுமுறைக் கோப்பில் தகவல்கள் தீர்ந்துவிட்டதா என்பதை EOF என்ற சார்பு கூறும். இதைப் பார்த்துத்தான் தகவல்களைப் பெற முயற்சிக்க வேண்டும். இந்தச் சார்பு நேரடி அணுகுமுறைக் கோப்புகளுக்குப் பொருந்தாது.

2. ஒரு வரிசை அணுகுமுறைக்கோப்பினை ஒரு சமயத்தில் ஒரு வழியில்தான் படிக்க அல்லது எழுத பயன்படுத்த முடியும். தேவைக்கேற்ப கோப்பினை மூடி மூடித் திறந்து கொள்ளவேண்டும்.

பயிற்சி வினாக்கள்

1. பல பொருட்களுக்கு, பொருளின் பெயர், அதை அளக்கும் குறியீடு (K கிலோ கிராம், L லிட்டர், U எண்ணிக்கை), இருப்பு அளவு முதலியவை கொண்ட ஒரு வரிசை அணுகுமுறை கோப்பினைத் தயார் செய்.
2. மேலே தயாரித்த கோப்பினைப் பயன்படுத்தி, கடையில் எத்தனை விதமான பொருட்கள் உள்ளன. அரிசி இருப்பு எவ்வளவு என்பனவற்றைக் கண்டுபிடி.
3. பலரின் பெயர்களும், வயதும் கொண்ட கோப்பு ஒன்றில் எத்தனை பெயர்கள் என்று பார். நூறு பேர்களுக்குக் குறைவாக இருந்தால், பகத்சிங், வயது:20, கட்டபொம்மன் வயது 18, சிதம்பரம், வயது : 18 என்னும் மூன்று பெயர்களையும், அந்தக் கோப்பின் இறுதியில் இணைத்து, கோப்பினை விரிவுபடுத்து. (நினைவிருக்கட்டும். இதற்கு கோப்பினை மூடிவிட்டு மறுமுறை திறக்க வேண்டும்.)
4. மேலே உள்ள கோப்பில், மதன் என்ற பெயர் இருந்தால், அந்தப் பெயரை மட்டும் (வயதுடன்) நீக்கிவிட்டு, ஒரு புதுக்கோப்பினைத் தயார் செய். இதற்கான போலிக்குறியீடு -

1. முதல் கோப்பினை படிப்பதற்கும், இரண்டாம் கோப்பினை எழுதுவதற்கும் திற.
2. (NOT EOF (முதல் கோப்பு))

   முதல் கோப்பிலிருந்து பெயர், வயதைப் படி

   (பெயர் <> மதன்) என்றால், வயதையும் இரண்டாம் கோப்பில் எழுது.
3. இரண்டு கோப்புகளையும் மூடு.

6.4 நேரடி அணுகுமுறைக் கோப்புகள்

நேரடி அணுகுமுறைக் கோப்புகளில், தகவல்கள் பதிவேடுகளாகத் தொகுக்கப்பட்டு, பதிவேடுகளாகவே வைக்கப்படும். எல்லாப் பதிவேடுகளும் ஒரே நீளமாக இருக்கும். அதனால் எந்தப் பதிவேட்டின் இடத்தையும் கணக்கிட்டு, அந்தப் பதிவேட்டை நேரடியாக எழுதலாம், படிக்கலாம் அல்லது புதியதாகச் சேர்க்கலாம். திறந்திருக்கும் கோப்பில் இந்த மூன்றையுமே நம் விருப்பம்போல் மாற்றி மாற்றிச் செய்யலாம். அதனால் இந்த கோப்புகளைத் திறக்கும்போது FOR INPUT, FOR OUTPUT, FOR APPEND என்று எதையும் கூறக்கூடாது. அதற்குப் பதிலாக ஒரு பதிவேட்டின் நீளம் எத்தனை பைட்டுகள் என்பதைக் கூறியாக வேண்டும்.

நேரடி அணுகுறைக் கோப்புகளைப் பயன்படுத்துவதற்கு முன்பு, அவற்றிலுள்ள பதிவேடுகளில் என்னென்ன தகவல்கள் இருக்கின்றன என்பதை வரையறை செய்துகொள்ளவேண்டும். ஒரு பதிவேட்டைப் படித்தால் அதிலிருந்து தகவல்களைச் சரியாகப் பிரித்து எடுக்கவேண்டும். இவற்றை எல்லாம் எவ்வாறு செய்வது என்பதை இந்தப் பாடத்தில் பார்ப்போம்.

பதிவேடுகளின் வரையறை

ஒரு பதிவேடு பல பைட்டுகளால் ஆனது. எத்தனை பைட்டுகள் என்பது LEN = record length என்பதன் மூலம் OPEN வாக்கியத்தில் குறிப்பிடப்படுகின்றது. ஒரு பதிவேடு பல தகவல்களின் தொகுப்பு. இந்தத் தகவல்கள் எல்லாமே சரங்களாக இருக்கவேண்டும். எண்களாக இருக்கக்கூடாது. என்னென்ன சரங்கள், அவை ஒவ்வொன்றும் எவ்வளவு நீளம், அவை எந்த வரிசையில் வருகின்றன என்பதை வரையறை செய்ய FIELD வாக்கியம் பயன்படுகிறது. இதன் இலக்கணம் -

FIELD # file number, width1 AS string variable1, width2 AS string variable2,...

உதாரணங்கள் :-

FILED # 1, 4 AS NUM$, 6 AS NAME$, 20 AS ADRESS$
FIELD # 2, 2 AS CODE$, 10 AS PHONE$

முதல் கோப்பில் மொத்தம் 30 பைட்டுகள். மூன்று சரங்கள். NUM$ என்பது முதல் 4 பைட்டுகள். 5 முதல் 10 வரை உள்ள பைட்டுகள் NAME$ ஐக் கொடுக்கும். 11 முதல் 30 வரை உள்ள பைட்டுகள் ADDRESS$ ஐக் கொடுக்கும்.

பலவித எண்கள்

சாதாரணமாக பேசிக்கில் ஒரு எண் 4 பைட்டுகளில் வைக்கப்பட்டிருக்கும். முழு எண், பின்னம் என்ற வேறுபாடு இல்லை. மேம்படுத்தப்பட்ட பேசிக்கில் எண்களை மூன்று விதங்களாகப் பிரிக்கலாம். -32,768 முதல் 32,767 வரை உள்ள முழு எண்கள் - இவற்றைக் குறிக்க இரண்டு பைட்டுகள் போதும். அதனால் தேவைப்படும் இடம் குறைவு. இவை **முழு எண்கள்** (integers) *என்ற வகை.*

அடுத்தது, 4 பைட்டுகளைப் பயன்படுத்தி, 6 தசம இலக்கங்கள் வரை சரியாகக் கொடுக்கும் பின்னங்கள். இவை மிதக்கும் புள்ளி முறையில் வைக்கப்படும். இது **ஒருமடங்கு துல்லியஎண்** (single precision number) *என்ற வகை.*

மூன்றாவது, 8 பைட்டுகளைப் பயன்படுத்தி, இன்னும் துல்லியமாக வைக்கப் படுவது. இது **இருமடங்கு துல்லியஎண்** (double precision number).

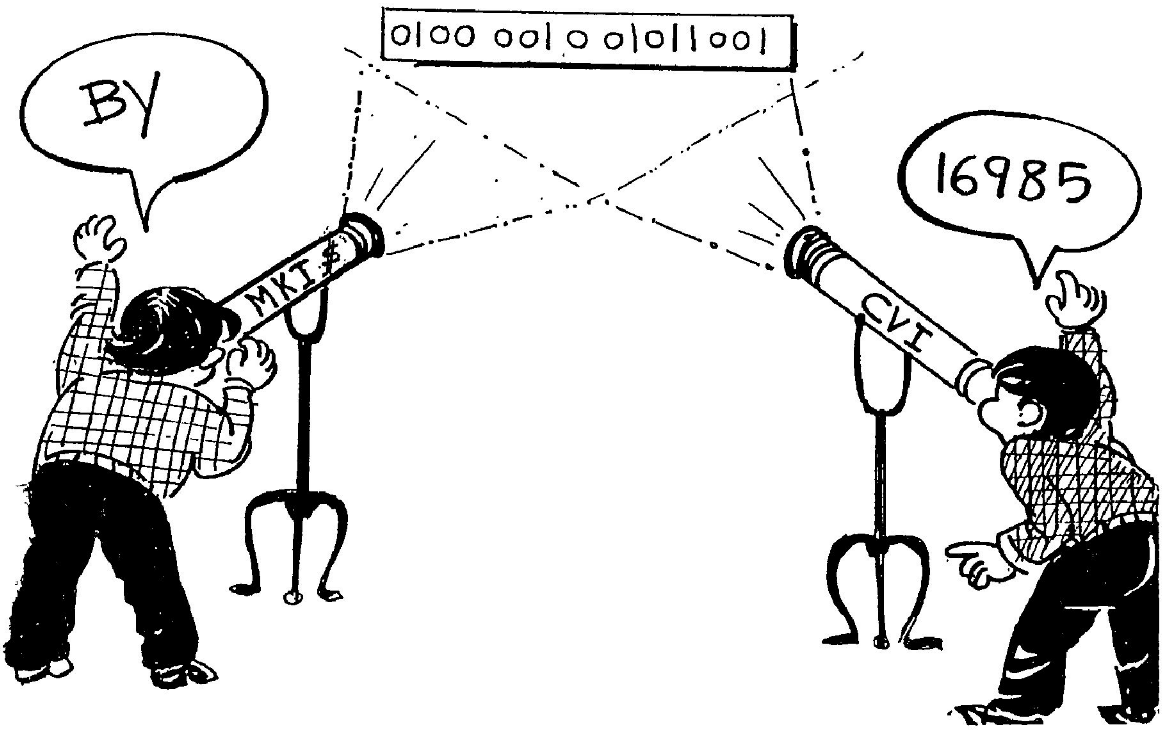
இந்த எண்களை எப்படி வேறுபடுத்திப் பார்ப்பது? இவற்றைக் குறிக்கும் மாறிகளை **முழுஎண் மாறி** (integer variable), **ஒருமடங்கு துல்லிய மாறி** (single precision variable), **இருமடங்கு துல்லிய மாறி** (double precision variable) *எனலாம். சரமாறியைக் குறிக்க பெயரின் கடைசியில் ஒரு டாலர் குறியீடு சேர்ப்பதைப் போல், இங்கும் செய்யலாம். பெயரின் இறுதியில் % சேர்த்தால் அது முழு எண் மாறி. # சேர்த்தால் அது இருமடங்கு துல்லியமாறி. ஒன்றும் சேர்க்கா விட்டால் அல்லது ! சேர்த்தால் அது ஒரு மடங்கு துல்லியமாறி.*

உதாரணங்கள்

A%, PAY%, MARKS% (7), RADIUS#, ANGLE#, DIFF#, A!, SINGLE!

பதிவேட்டைத் தொகுக்கும் முறை

ஒரு பதிவேடு பல சரங்களால் ஆனது என்று பார்த்தோம். எண்களையும், பதிவேட்டில் வைக்க வேண்டியது அவசியம் என்பது தெரிந்த விஷயம். அப்படியானால் எண்களை எவ்வாறு பதிவேட்டில் வைப்பது? எண்களையும் சரங்களாக மாற்றித்தான் வைக்க வேண்டும். ஒரு பின்னம் சாதாரணமாக 4 பைட்டுகளில் இருக்கும். இதை வேறொன்றும் செய்யத் தேவையில்லை. 4 பைட்டுகள் கொண்ட சரமாக, அதாவது 4 ஆஸ்கி குறியீடுகள் கொண்ட சரமாகப் பார்த்தால் போதும். இதே போல், முழு எண்ணை 2 பைட் நீள சரமாகவும், இருமடங்கு துல்லிய எண்ணை 8 பைட்டுகள் நீள சரமாகவும் பார்க்க வேண்டும். தகவல்களில் மாறுதல் எதுவுமில்லை. அவற்றை எப்படிப் பார்க்கிறோம் என்ற பார்வையில்தான் வேறுபாடு. அழகு என்பது கூட பார்ப்பவர் பார்வையில்தானே இருக்கிறது!



''இது வழியாப் பாத்தா ரெண்டு ஆஸ்கி குறியீடு தெரியுது.''
''இது வழியாப் பாத்தா ஒரு முழு எண் தெரியுது.''

எண்களைச் சரங்களாக மாற்றித் தர மூன்று சார்புகள் உள்ளன. உண்மையில் இவை எதையும் கணிப்பதில்லை. எந்த வேலையும் செய்வதில்லை. அவற்றிடம் கொடுத்த எண்களை அப்படியே திருப்பிக் கொடுத்து, ''இவை சரங்கள்'' என்று கூறிவிடும். இவை make எனப்படும் சார்புகள். வரும் மதிப்பு சரமானதால், பெயரில் $ குறியீடு வருவதைக் கவனிக்கவும். இவை--

MKI$ (integer expression),

MKS$ (single precision expression),

MKD$ (double precision expression)

உதாரணமாக, MKI$ (4 + 4876), MKS$ (8246.73), MKD$ (2.3E - 42) என்பவை முறையே 2, 4, 8 பைட்டுகள் கொண்ட சரங்களைக் கொடுக்கும்.

சரங்களை முதலில் இடைஇடத்தில் எழுதி, ஒரு பதிவேட்டை உருவாக்க வேண்டும். பிறகு அந்தப் பதிவேட்டை இடைஇடத்தில் இருந்து கோப்புக்கு மாற்ற வேண்டும். நேரடியாக கோப்பில் எழுத முடியாது. இடைஇடத்தில் சரங்களை எழுத LSET, RSET (left set, right set) என இரு சிறப்பு வாக்கியங்கள் இருக்கின்றன. இவற்றை விட்டு விட்டு; LET, INPUT போன்ற வாக்கியங்களைப் பயன்படுத்தக் கூடாது. LEST, RSET என்பவற்றின் இலக்கணம், LET என்பதற்கான இலக்கணம் போலவேதான்.

LSET string variable = string expression

RSET string variable = string expression

LSET மற்றும் RSET செயல்படும் விதமானது, சரமாறிக்கு பதிவேட்டில் ஒதுக்கப் பட்டுள்ள அளவையும், அதில் எழுத வேண்டிய சரத்தின் அளவையும் பொறுத்தது.

1. சரமாறியின் அளவும், வரும் சரத்தின் அளவும் சமம் என்றால், இரண்டுமே முழுச் சரத்தை அப்படியே எழுதும்.

2. சரமாறியின் அளவு, வரும் சரத்தின் அளவை விடக் குறைவு என்றால், இரண்டுமே, சரத்தின் இடது பக்கத்திலிருந்து எழுத ஆரம்பிக்கும், சரத்தின் வலது கோடியில் உள்ள சில குறியீடுகளுக்கு இடம் இல்லாமல் போய்விடும். இதிலும் RSET, LSET இரண்டுமே ஒரே விதமாக வேலை செய்யும்.

3. சரமாறியின் அளவு, வரும் சரத்தின் அளவை விட அதிகமாக இருந்தால், இரண்டிற்கும் வித்தியாசம் உண்டு. LSET என்பது மாறியின் இடத்தில் இடது புறமாக சரத்தை வைத்துவிட்டு, வலதுபுறம் மீதி இருக்கும் இடங்களை இடைவெளிகளால் நிரப்பும். RSET என்பது மாறியின் இடத்தில் வலதுபுறமாக சரத்தை வைத்துவிட்டு, இடதுபுறம் மீதி இருக்கும் இடங்களை இடைவெளிகளால் நிரப்பும்.

சில உதாரணங்கள்

FIELD #1,4 AS A$, 6 AS B$

X$ = "GOOD" : Y$ = "Y" : Z$ = "BANGALORE"

என இருக்கட்டும். இப்போது சில LSET, RSET வாக்கியங்களையும், அவை கொடுக்கும் மதிப்புகளையும் பார்ப்போம்.

LSET A$ = X$	A$ என்பது GOOD என ஆகும்
LSET A$ = Y$	A$ என்பது Y "
LSET A$ = Z$	A$ என்பது BANG "
RSET B$ = X$	B$ என்பது b̸b̸GOOD "
RSET B$ = Y$	B$ என்பது b̸ b̸b̸b̸b̸Y "
RSET B$ = Z$	B$ என்பது BANGAL "

இவ்வாறு இடைஇடத்தில் உள்ள எல்லாச் சரங்களிலும் எழுதியவுடன் ஒரு பதிவேடு பூர்த்தியாகிறது. இந்தப் பதிவேட்டை அப்படியே கோப்பிற்கு மாற்ற வேண்டும். இதற்கு, இந்தப் பதிவேடு எந்தக் கோப்பில் எத்தனையாவது பதிவேடாக வைக்கப்பட வேண்டும் என்பதைக் கூற வேண்டும். இதைச் செய்யும் PUT என்ற வாக்கியத்தின் இலக்கணம்

PUT# file number, record number

ஒரு கோப்பில் அதிக பட்சம் *32,767* பதிவேடுகள் இருக்கலாம். அதனால் record number *1* முதல் *32,767* வரை இருக்கலாம். இது பின்னமாக இருந்தால், அருகிலுள்ள முழு எண்ணாக மாற்றப்பட்டு விடும்.

உதாரணமாக, PUT # 1, 7

PUT # 2, 2345.7

என்று எழுதலாம்.

கோப்பிலிருந்து படிக்க

கோப்பிலிருந்து தகவல்களைப் பெற, முதலில் தேவையான பதிவேட்டை கோப்பிலிருந்து இடைஇடத்திற்கு மாற்ற வேண்டும். பிறகு இடைஇடத்திலிருந்து சரங்களை எடுத்துப் பயன்படுத்த வேண்டும்.

குறிப்பிட்ட எண்ணுள்ள கோப்பிலிருந்து, குறிப்பிட்ட எண்ணுள்ள பதிவேட்டை எடுத்து, அந்தக் கோப்பிற்கான இடைஇடத்தில் வைப்பதற்கு GET என்னும் வாக்கியம் பயன்படுகிறது.

இதன் இலக்கணம், GET # file number, record number

உதாரணங்கள்- GET # 3, 6002

GET # 1, 42.78 + 2.1

வைக்கப்பட்ட தகவல்கள் சரங்கள் என்றால் அப்படியே எடுத்துப் பயன்படுத்திக் கொள்ளலாம். ஒரு எண்ணைச் சரமாக மாற்றிப் பதிவேட்டில் வைத்திருந்தால், அந்தச் சரத்தைத் திரும்பவும் ஒரு எண்ணாக மாற்றியபிறகே சரியான முறையில் பயன்படுத்த முடியும். இவ்வாறு சரத்தை எண்ணாக மாற்ற (convert) CVI, CVS, CVD என்ற மூன்று சார்புகள் உள்ளன. CVI (2 byte string) என்பது ஒரு முழு எண்ணையும் *(2 பைட்)*, CVS (4 byte string) என்பது ஒரு சாதாரண *(4 பைட்)* எண்ணையும், CVD (8 byte string) என்பது ஒரு இருமடங்கு துல்லிய *(8 பைட்)* எண்ணையும் கொடுக்கும்.

உதாரணங்கள் - TOTAL% = CVI (PAY$) + 35

LENGTH = CVS(L$) + 72.88

PRECISE # = CVD (PRES$)

நினைவில் நிற்க

1. நேரடி அணுகுமுறைக் கோப்பில், பதிவேடுகளாக எழுத, படிக்கவேண்டும்.
2. பதிவேடுகளில் எண்களையும் சரங்களாக மாற்றித்தான் வைக்கவேண்டும். அதற்கு MKI$, MKS$, MKD$ என்ற சார்புகள் பயன்படும். சரங்களை மீண்டும் எண்களாக மாற்ற CVI, CVS, CVD என்ற சார்புகள் பயன்படும்.
3. இடைஇடத்தில் சரங்களை எழுத LSET, RSET பயன்படும். LET, INPUT பயன்படுத்தக் கூடாது.
4. திறந்துள்ள கோப்பில் மாற்றி மாற்றிப் படிக்கலாம் அல்லது எழுதலாம்.

பயிற்சி வினாக்கள்

1. ஒரு பதிவேட்டை எவ்வாறு வரையறை செய்வது என்பதை உதாரணங்களுடன் விளக்கு.
2. LEN என்பதன் பயன்பாடு என்ன?
3. LSET, RSETஎவ்வாறு எங்கு பயன்படுகின்றன?
4. LSETஐ RSET உடன் ஒப்பிடு.
5. பதிவேடுகளை இடைஇடத்திலிருந்து கோப்பிற்கும், கோப்பிலிருந்து இடை இடத்திற்கும் மாற்றுவது எவ்வாறு?

6.5 நேரடி அணுகுமுறை உதாரணங்கள்

ஒரு கோப்பில் உள்ள தகவல்களில் சிலவற்றை மட்டும் மாற்ற வேண்டுமென்றால், வரிசை அணுகுமுறையில், ஒரு புதிய கோப்பினையே உருவாக்க வேண்டும். ஆனால் நேரடி அணுகுமுறையில் இதை எளிதாகச் செய்யலாம். தேவையான பதிவேடுகளை மட்டும் படித்து, அவற்றில் மாற்றங்கள் செய்து மீண்டும் அவற்றின் இடங்களிலேயே வைத்துவிடலாம். முழுக் கோப்பினையும் படிக்க வேண்டியதில்லை. புதிய கோப்பினை உருவாக்க வேண்டியதில்லை. நேரடி அணுகுமுறையில் கோப்புகளைப் பயன்படுத்த சில உதாரணங்களை இங்கு பார்ப்போம்.

1. ஒரு வங்கியில் சேமிப்புக் கணக்கு வாடிக்கையாளர்களுக்கு 1, 2... என்று வரிசையாக எண்கள் கொடுக்கப்பட்டுள்ளன. ஒவ்வொரு எண்ணுக்கும் ஒரு பதிவேடு உள்ள ஒரு நேரடி அணுகுமுறைக் கோப்பு உள்ளது. இந்தப் பதிவேட்டில், முக்கிய விவரங்களான கணக்கெண் (2 பைட்டுகள்--முழுஎண்), பெயர் (20பைட்டுகள்), இருப்புத் தொகை (4 பைட்டுகள்) என்பவை மட்டும் உள்ளன. ஒவ்வொரு முறை பணம் போடும்போதும், எடுக்கும்போதும், இந்தத் தொகையை மாற்ற வேண்டும். இதைச் செயல்படுத்த ஒரு கட்டளை நிரல் எழுது.

1. கோப்பினைத் திற.
2. பதிவேட்டை வரையறை செய்.
3. திரும்பத் திரும்ப செய்

    பணம் போட வா, எடுக்கவா, கணிப்பு முடிந்ததா என்று கேள்

    பணம் போட என்றால், இருப்பினைக் கூட்டிவை.

    பணம் எடுக்க என்றால், தேவையான பணம் இருந்தால் அதைக் கழி:

    இல்லையென்றால் அதைச் சொல்.

    முடிவு என்றால் வளையத்தை விட்டு வெளியே செல்.

4. கோப்பினை மூடு.

```
10  OPEN "SAVE.DAT" AS #1 LEN=26
20  FIELD #1, 2 AS NUM$, 20 AS NAME$, 4 AS BALANCE$
30  REM
100 WHILE 1=1
110     PRINT "D FOR DEPOSIT, W FOR WITH DRAWAL, Q FOR
        QUITTING"
120     I NPUT "YOUR CHOICE:", CHOICE$
```

```
130      IF CHOICE$  =  "D"  GO SUB 1000  : REM DEPOSIT
140      IF CHOICE$  =  "W"  GO SUB 2000: REM WITHDRAW
150      IF CHOICE$  =  "Q" GOTO 180
160 WEND
170  CLOSE #1
180  END
1000 REM SUBROUTINE FOR DEPOSIT
1010 INPUT "GIVE A/C NO.& AMOUNT", ACNO%, AMT
1020 GET #1, ACNO%
1030 NEW = CVS (BALANCE$) + AMT
1040 LSET BALANCE$ = MKS$ (NEW)
1050 PUT #1, ACNO%
1060  RETURN
1070 REM END OF DEPOSIT
2000 REM SUBROUTINE FOR WITHDRAWAL
2010 INPUT "GIVE A/C NO.& AMOUNT",  AC NO% ,   AMT
2020 GET#, ACNO%
2030 IF AMT > CVS (BALANCE$) THEN  2040 ELSE 2100
2040 PRINT "NOT ENOUGH MONEY AVAILABLE. SORRY"
2050  RETURN
2100  REM MONEY AVAILABLE
2110  NEW = CVS (BALANCE$) - AMT
2120  LSET BALANCE$ = MKS$ (NEW)
2130  PUT #1, ACNO%
2140  RETURN
2150  REM END OF WITHDRAWAL
```

இங்கு முடிவில்லாமல் திரும்பத்திரும்பச் செய்வதற்கு WHILE 1 = 1 பயன்படுத்தப் பட்டுள்ளது. இரண்டு துணை முறைகளைப் பயன்படுத்துவதனால் முதன்மை நிரல் எளிமையாகிறது.

2. ஒரு நிறுவனத்தில் 200 பணியாளர்கள் உள்ளனர். அவர்களுக்கு 1 முதல் 200 வரை அடையாள எண்கள் கொடுக்கப்பட்டுள்ளன. கோப்பில் அவர்களுடைய பெயர் (20 பைட்டுகள்), பணியாளர் எண்(2 பைட்டுகள்), வேலையில் சேர்ந்த வருடம் (2 பைட்டுகள்), அடிப்படைச் சம்பளம் (4பைட்டுகள்) என்ற தகவல்கள் உள்ளன. சில

பணியாளர்களுக்கு அடிப்படைச் சம்பளத்தில் 10 விழுக்காடு உயர்வு தர வேண்டும். இந்தப் பட்டியல், அந்தப் பணியாளர் பெயர் மற்றும் எண்ணுடன் கையில் இருக்கிறது. கோப்பில் தேவையான மாற்றங்களைச் செய்ய ஒரு கட்டளை நிரல் எழுது. மாற்றம் செய்யும் பதிவேடுகளில் பெயர்கள் சரியாக இருக்கின்றனவா என்று சரிபார்த்துக் கொள்.

1. கோப்பினைத்திற

2. பதிவேட்டை வரையறை செய்.

3. இன்னும் = ''ஆம்''

4. (இன்னும் = ''ஆம்'') என்று இருக்கும் வரை செய்

பணியாளர் எண்ணைப் பெறு.

அந்தப் பதிவேட்டை எடு.

பெயரைத் திரையில் காண்பித்து, சரியா என்று கேள்.

சரி என்றால், சம்பளத்தை மாற்றி, பதிவேட்டைத் திரும்ப வை.

இன்னும் செய்ய வேண்டுமா என்று கேட்டுப் பெறு.

5. கோப்பினை மூடு.

```
10 OPEN  "EMP. DAT" AS #2 LEN  = 28
20 FIELD #2, 20 AS NAME$, 2 AS NUM$, 2 AS YEAR$, 4 AS PAY$
30 REM
40 MORE$ = "Y"
50 WHILE MORE$ = "Y"
60      INPUT "GIVE EMPLOYEE NUMBER", NUMBER%
70      GET#2, NUMBER%
80      PRINT "NAME IS", NAME$
90      PRINT "IS IT CORRECT? SAY Y OR N", COR$
100     IF COR$ = "Y" THEN 110 ELSE 150
110                 BASIC = CVS (PAY$) *1.1
120                 LSET PAY$ = MKS$ (BASIC)
130                 PUT # 2 ,  NUMBER%
140     REM
150     PRINT "ANY MORE? SAY Y OR N", MORE$
160 WEND
200 CLOSE # 2
210 END
```

3. மேலே பார்த்த கணக்கில், எல்லோருக்குமே ஊதிய உயர்வு கொடுக்கப்பட வேண்டும். ஆயிரம் ரூபாய்க்குக் குறைவாக உள்ளவர்களுக்கு 10 விழுக்காடும் மற்றவர்களுக்கு 8 விழுக்காடும் உயர்வு அளிக்க வேண்டும் என்று மாற்றி அமை.

நேரடி அணுகுமுறைக் கோப்பாக இருந்தாலும், இந்தக் கணிப்பிற்கு , வரிசையாக 1, 2 ....என்ற எண்கள் உள்ள பதிவேடுகளை எடுத்து, அவற்றை மாற்றி அமைப்பதே சிறந்த முறை. கோப்பில் மாற்றம் எதுவும் இல்லை. நேரடி அணுகுமுறையிலேயே திறந்து, பயன்படுத்த வேண்டும்.

```
10  OPEN "EMP. DAT" AS #2 LEN = 28
20  FIELD # 2, 20 AS NAME$, 2 AS NUM$, 2 AS YEAR$, 4 AS PAY$
30  REM
40  FOR I = 1 TO 200
50          GET# 2, I
60          BASIC = CVS (PAY$)
70          IF BASIC < 1000 THEN BASIC = BASIC*1.1
                    ELSE BASIC = BASIC *1.08
80          LSET PAY$ = MKS$ (BASIC)
90          PUT # 2, I
100         NEXT I
110 CLOSE #2
120 END
```

4. ஒரு வகுப்பிலுள்ள மாணவர்களின் பெயர், வகுப்பு எண், ஐந்து மதிப்பெண்கள் அடங்கிய ஒரு நேரடி அணுகுமுறைக் கோப்பினைத் தயார் செய்.

கோப்பினை உருவாக்க புதிதாக எதுவும் கற்றுக் கொள்ள வேண்டியதில்லை. பதிவேடுகளை உருவாக்கி ஒவ்வொன்றாக எழுத வேண்டும். அவ்வளவுதான். கணிப்பொறியே தேவையான இடத்தைத் தகடுகளில் ஒதுக்கிக் கொள்ளும்.

1. கோப்பினைத் திற.
2. பதிவேட்டினை வரையறை செய்.
3. இன்னும் = ''ஆம்''
4. (இன்னும் = ''ஆம்'') என்று இருக்கும் வரை செய்.
   - ஒரு பதிவேட்டின் தகவல்களை கேட்டு வாங்கு.
   - பதிவேட்டை உருவாக்கு.
   - பதிவேட்டை கோப்பில் எழுது.
   - இன்னும் செய்ய வேண்டுமா என்று கேட்டு வாங்கு
5. கோப்பினை மூடு.

```
10 OPEN "MARKS.DAT" AS # 3 LEN = 32
20 FIELD # 3, 20 AS NAME$, 2 AS ROLLNO$, 2 AS MKI$
        2 AS MK2$, 2 AS MK3$, 2 AS MK4$, 2 AS MK4$, 2 AS MK5$
30 CONT$ = "Y"
40 WHILE CONT$ = "Y"
50      INPUT "NAME"; N$
60      INPUT "ROLL NO:"; R%
70      INPUT "FIVE MARKS : " ; MI%, M2%, M3%, M4%, M5%
80      REM
90      LSET NAME$ = N$
100     LSET ROLLNO$ = MKI$(R%)
110     LSET MK1$ = MKI$(M1%)
120     LSET MK2$ = MKI$(M2%)
130     LSET MK3$ = MKI$(M3%)
140     LSET MK4$ = MKI$(M4%)
150     LSET MK5$ = MKI$(M5%)
160     REM
170     PUT# 3, R%
180     INPUT "ANY MORY? Y OR N"; CONT$
190 WEND
200 CLOSE # 3
210 END
```

நினைவில் நிற்க

1. முதலில் தகவலை INPUT, READ மூலம் பெற்று, பிறகு LSET, RSET மூலம் தான் பதிவேடுகளை உருவாக்க வேண்டும். நேரடியாக INPUT மூலம் உருவாக்கக் கூடாது.

2. EOF என்பதை நேரடி அணுகுமுறைக் கோப்பிற்குப் பயன்படுத்தக் கூடாது.

பயிற்சி வினாக்கள்

1. ஒரு நூலகத்தில் சுமார் 10,000 நூல்கள் உள்ளன. புத்தகத்தின் பெயர், வரிசை எண், ஆசிரியர் பெயர், விலை, பதிப்பகத்தின் பெயர், வெளியிட்ட வருடம், புத்தகம்

நூலகத்தில் இருக்கிறதா அல்லது இரவல் கொடுக்கப்பட்டுள்ளதா என்ற தகவல், இரவல் கொடுக்கப்பட்டிருந்தால், புத்தகம் எடுத்தவருடைய எண், என்ற தகவல்கள் அடங்கிய நேரடி அணுகுமுறைக் கோப்பு ஒன்று உள்ளது. இந்தக் கோப்பினைப் பயன்படுத்தி, கீழே உள்ள ஒவ்வொரு செயலுக்கும் ஒரு தனி கட்டளை நிரல் எழுது. முதலில் போலிக் குறியீட்டில் எழுது.

அ. வரிசை எண்கள் 2046, 17, 132, 1066 என்று உள்ள புத்தகங்களின் தகவல்கள் தவறாக எழுதப்பட்டுள்ளன. அவற்றைத் திருத்தி எழுத வேண்டும்.

ஆ. சுஜாதா எழுதிய நூல்களில் இந்த நூலகத்தில் இருப்பவற்றின் பட்டியலைத் தயார் செய். (இதற்கு வரிசை முறையில் அணுக வேண்டும். EOF பயன்படுத்தக் கூடாது.)

இ. நூலகத்தில் ''புதிர் , சிந்தனை, கணிதம் '' என்ற நூல் இருக்கிறதா? என்று பார்.

ஈ. 1989 முதல் 1991 வரை வெளியிடப்பட்ட, எல்லா நூல்களையும் பட்டியலிடு.

2. சில புதிய நூல்களைக் கோப்பில் சேர்க்க வேண்டும். இதுவரை 10,540 நூல்கள் ஏற்கனவே உள்ளன. நூலகத்தில் உள்ள நூல்களின் எண்ணிக்கை, அவற்றின் மொத்த மதிப்பு, இரண்டையும் கண்டுபிடி.

3. ஒரு நேரடி அணுகுமுறைக் கோப்பில் பெயர், எண் (1 முதல் 2000 வரை), சேர விரும்பும் படிப்பின் பெயர் (MCA அல்லது DCA) முதலிய தகவல்கள் உள்ள பதிவேடுகள் உள்ளன. இதில் ஏற்கனவே உள்ள எண்க்ளைப் பின்வருமாறு மாற்றி அமைக்க வேண்டும். MCA என்றால் முன்னால் 1 என்றும், DCA என்றால் 2 என்றும் சேர்க்க வேண்டும். உதாரணமாக எண் 630, MCA என்றால் புது எண் 1630; எண் 1288, DCA என்றால் 21288 என்பது புது எண்.

7

கணிப்பொறி வரைவியல்

7 - 1 இன்றைய தேவை

7 - 2 அடிப்படைச் செய்திகள்

7 - 3 திரை, வண்ணத் தேர்வு

7 - 4 புள்ளிகள்

7 - 5 நேர்கோடும் செவ்வகமும்

7 - 6 தொடர்ந்து வரைவது

7 - 7 வட்டம்

7 - 8 சலனப் படங்கள்

7 - 9 எழுத்துத் திரையும் ஓசையும்

எண்களைக் கொண்டு கணிக்கும் இயந்திரமாகத் தோன்றிய கணிப்பொறி, படிப்படியாக முன்னேறி இன்று தொலைக்காட்சிப் பெட்டிகளுடன் போட்டி போடும் அளவில், படங்கள் பலவற்றையும் திறம்பட வரைந்து காட்டும் திறன் பெற்றுள்ளது. கணிப்பொறி வரைவியலுக்கு ஓர் அறிமுகத்தைப் பெறுவது இந்தப் பகுதியின் நோக்கம். கோடுகள், வட்டங்கள், நீள்வட்டங்கள் இவற்றைக் கொண்டு சிறு படங்களை எப்படி வரைவது என்று பார்ப்போம். அவற்றை எவ்வாறு நகரச் செய்து சலனப் படங்களை உருவாக்கலாம் என்றும் பார்ப்போம். படங்கள் வரைவதற்கு முன்னால் தேவையான நுட்பத்தில் திரையை எவ்வாறு தயார் செய்வது என்பதையும் காண்போம்.

7.1 இன்றைய தேவை

கணிப்பொறி தோன்றிய காலத்தில் அது எண்களைக் கொண்டு கணிக்கும் கருவியாகவே நினைக்கப்பட்டது. படங்களைத் திரையில் காண்பிப்பதும், அவற்றைச் சலனப்படங்களாக்கி, கணிப்பொறியின் திரையைத் திரைப்படத்திரை போல் மாற்றி அமைப்பதும் பிற்காலத்தில் ஏற்பட்ட மாறுதல்கள். இவை இன்று பல துறைகளிலும் பயன்படுகின்றன. சில உதாரணங்களைப் பார்ப்போம்.

விவரங்களை எண்களாகப் பார்க்காமல், வரைபடங்களாகப் பார்க்கும்போது அவை மனத்தில் எளிதில் பதிகின்றன. அதனால், சாதாரண கோடுகளால் ஆன வரைபடங்கள், **பட்டை வரைபடம்** (bar chart), **வட்ட வரைபடம்** (pie chart), போன்றவை வணிகத்துறையில் பெரிதும் பயன்படுகின்றன. விளம்பரத்துறைக்கும், திரைப்படத் துறைக்கும், கவர்ச்சி மிக்க விளம்பரங்களை உருவாக்குதல், அழகிய எழுத்துகளைப் பல வடிவங்களில் உருவாக்குதல், உண்மையில் இல்லாத உருவங்களையும், நடக்க முடியாத செயல்களையும் திரையில் காண்பித்தல், வண்ணங்களை மாற்றுதல் போன்ற பல செயல்கள் மூலம் கணிப்பொறி பயன்படுகிறது. ஒரு உருவம் சிறிது சிறிதாக மாறி வேறொரு உருவமாக வடிவெடுக்கும் **உருமாற்றம்** (morphing) என்பது இன்றைய புது தொழில் நுட்பம்.

மருத்துவத் துறையில் இன்று படங்கள் பெரிதும் பயனாகின்றன. மீயொலி அலைகள், மின்காந்த அலைகள் போன்றவை மூலம் உடலின் உள் உறுப்புகளை ஆராயும்போது, அப்பாகங்களின் முப்பரிமாணத் தோற்றங்கள் திரையில் தெரிகின்றன. இதனால் உடலின் உள் உறுப்புகளைத் தெளிவாகப் பார்த்து ஆராய முடிகிறது, ஒரு துளி ரத்தமும் சிந்தாமலேயே ! பொருள்களை வடிவமைப்பதில், அப்பொருள்களின் தோற்றங்களைத் திரையில் காண்பிப்பதில் கணிப்பொறி பெரிதும் பயன்படுகிறது. இதனால், ஏராளமான பொருட் செலவில் மாதிரிகளைத் தயாரிப்பது தவிர்க்கப்படுகிறது. நேரமும் மிச்சமாகிறது.

பார்ப்பவர் கண்களுக்கு மட்டும், ஒரு முப்பரிமாண மாய உலகைக் காட்டி, அவர் அந்த மாய உலகத்தில் இருந்துகொண்டு செயல்படுவது போன்ற தோற்றத்தை ஏற்படுத்தும் **மாயத் தோற்றம்** (Virtual reality) என்ற புதிய தொழில் நுட்பத்தின் அடிப்படை வரைவியல்தான்.

சாதாரண மற்றும் சலனப்படங்களைக் கையாளுவதில் திறமை பெற்றுவிட்ட இந்த நேரத்தில், ஒலி, ஒளி இரண்டையும் கணிப்பொறியுடன் இணைத்து, **பல்லூடகம்** (multimedia) என்ற தொழில் நுட்பம் உருவாக்கப்பட்டு வளர்ந்து வருகிறது. இது கணிப்பொறியின் கட்டுப்பாட்டில் ஒரு **ஒளிப்பேழை** (video casettee) இயங்குவது போன்ற தோற்றத்தை அளிக்கும். தகவல்களை நம் விருப்பப்படி திரையில் காட்டலாம். அதைப் போலவே சலனப்படங்களையும் தேவைக்கேற்றபடி திரையில்

காண்பிக்க முடியும். இனி வருங்காலத்தில் கணிப்பொறி, தொலைக்காட்சி, தொலைபேசி, ஒலி, ஒளிப் பேழைகள் போன்ற எல்லாமே ஒன்று சேர்ந்து ஒருங்கிணைந்த தொழில் நுட்பமாக உருவாகும். அதன் முதல்படிதான் பல்ஊடகம். இப்போது காகிதத்தில் அச்சடிக்கப்பட்ட புத்தகங்களைப் படித்துக் கொண்டிருக்கிறோம். இன்னும் சில ஆண்டுகளில், இந்த பல்ஊடகத்தில் உருவாக்கப்படும் புத்தகங்களைப் படிப்போம். தேவையான பக்கங்களை, வேண்டும் விளக்கங்களை, சலனப்படங்களாகப் பார்த்துப் புரிந்து கொள்ளலாம். கற்றுக் கொள்வது ஒரு இனிய அனுபவமாகிவிடும் அப்போது. கணிப்பொறியைப் பயன்படுத்தும் எவருக்கும் அதில் படங்கள் வரைந்து பார்க்கத் தோன்றுவது இயற்கை. பேசிக் மொழியிலும் படங்கள் வரையத் தேவையான வசதிகள் உண்டு. அவற்றைப் பற்றிய ஓர் அறிமுகத்தை இங்கு காண்போம். முதலில், படங்கள் வரையத் தேவையான சாதனங்கள் எவை என்று பார்ப்போம். திரையை படம் வரையத் தயார் செய்யும் முறையை அடுத்ததாகப் பார்ப்போம். மூன்றாவதாக, சில சிறுபடங்களை உருவாக்கத் தேவையான அடிப்படைக் கட்டளைகளை அறிந்துகொள்வோம். அடுத்து படங்களை நகரவைக்கும் எளிய முறையைக் கற்றுக் கொள்வோம். படங்கள் நகரும்போது பின்னணியில் இசை இருந்தால் சிறப்பாக இருக்குமல்லவா? அதற்கான கட்டளை ஒன்றினைப் பார்ப்பதுடன் இப்பகுதியை நிறைவு செய்வோம்.

நினைவில் நிற்க

1. உருமாற்றம், பல்ஊடகம் போன்றவை தற்போது வளர்ந்துவரும் தொழில் நுட்பங்கள், கணிப்பொறி வரைவியல் இவற்றுக்கு அடிப்படை.

பயிற்சி வினா

1. கணிப்பொறி வரைவியல் பயன்படும் இடங்கள் நான்கினைக் கூறு.

7.2 அடிப்படைச் செய்திகள்

தேவையான வன்பொருட்கள்

படங்கள் வரையத் தேவைப்படும் வன்பொருட்கள் இரண்டு - **திரைச்சாதனம்** (monitor), மற்றும் அந்தத் திரையைக் கட்டுப்படுத்தும் அட்டை (I/O card). திரைகளில் இரு வகைகள் உள்ளன. ஒருவண்ணத் திரைகள் மற்றும் பலவண்ணத் திரைகள். ஒருவண்ணத்திரையில் வெற்றுத்திரை கறுப்புவண்ணத்தில் இருக்கும். இதில் ஏதாவது ஒரு வண்ணத்தில் எழுதலாம் அல்லது வரையலாம். பச்சை வண்ணத்தில் எழுதும் திரைகள் முதலிலும், பிறகு ஆரஞ்சு வண்ணத் திரைகளும் வந்து போயின. இப்போது வெண்பழுப்பு (soft white) நிறத் திரைகள் பிரபலமாக உள்ளன. இவை மலிவான திரைச் சாதனங்கள், பலவண்ணத் திரைகள் விலை அதிகமானவை.

திரையில் ஒரு சமயத்தில் மொத்தம் எத்தனை புள்ளிகள் இருக்கின்றன என்பதைப் பொறுத்து, இவற்றைப் பல வகைகளாகப் பிரிக்கலாம். புள்ளிகளின் எண்ணிக்கையைப் பொறுத்து **பிரிப்பாற்றல்** (resolution) மாறுபடுகிறது. பிரிப்பாற்றலுக்கு ஏற்றவாறு திரைச் சாதனமும் அதைக் கட்டுப்படுத்தும் அட்டையும் வேறுபடும். சில முக்கிய பிரிப்பாற்றல் வகைகளை இங்கு காண்போம்.

இங்கு முதலில் காண வேண்டியது CGA (color graphics Adaptor) என்னும் அட்டை. இது **நடுத்தரப் பிரிப்பாற்றல்** (medium resolution), **உயர் பிரிப்பாற்றல்** (high resolution) என்று இருவிதங்களில் செயல்படும். உயர்பிரிப்பில், ஒரு வரிசைக்கு 640 புள்ளிகளாக, 200 வரிசைகள் இருக்கும். ஆக மொத்தம் 640 X 200=1,28,000 புள்ளிகள் ! இவ்வளவா என்று மலைக்க வேண்டாம். இனி வரப்போவதைப் பாருங்கள் ! இந்தப் பிரிப்பாற்றலில் கறுப்பு, வெள்ளையில் படங்கள் வரைய முடியும். ஒரு புள்ளியில் 0 என்றால் கறுப்பு, 1 என்றால் வெள்ளை என்று வரையும். அதாவது ஒரு புள்ளியின் வண்ணத்தைக் குறிக்க ஒரு பிட் போதும். அதனால் திரையை முழுவதும் குறிப்பிட 1,28,000/8 = 16K பைட்டுகள் தேவை.

நடுத்தரப் பிரிப்பாற்றலில் 320 X 200 என்ற அளவில் புள்ளிகள் இருக்கும். ஒரு புள்ளியில் நாலு வண்ணங்களில் ஒன்றினைத் தேர்ந்தெடுக்கலாம். 00, 01, 10, 11 என்று நான்கு எண்களைக் குறிப்பிட இரண்டு பிட்டுகள் தேவை. அதனால், மொத்தம் $(320 \times 200 \times 2)/8 = 16$ K பைட்டுகள் தேவை. ஆகவே இந்த பிரிப்பாற்றல்கள் இரண்டுக்குமே ஒரே அளவில் தான் நினைவிடம் தேவைப்படுகிறது.

மேலே பார்த்தவை வரைபடத்திரைகள். எழுத்துத் திரையாகப் பயன்படுத்தும் போது வரிக்கு 80 குறியீடுகள் அல்லது 40 குறியீடுகள் என மொத்தம் 25 வரிகள் வரும். வரிக்கு 80 குறியீடுகள் என்றால் ஒரு குறியீட்டிற்கு 8×8 புள்ளிகள் என்ற அளவில் இடம் தேவைப்படும்.

வரிக்கு 40 குறியீடுகள் இருந்தால் 8×16 புள்ளிகள் தேவைப்படும்.

அடுத்தது EGA(Enchnced Graphic Adaptor) அட்டை. இது 640×350, 640×200 பிரிப்பாற்றல்களில் 16 வண்ணங்களில் செயல்படும்.

அடுத்தது VGA (Video Graphics Array-Aஎன்பது Adaptorஇல்லை). இது 640×480, 640×350, 640×200 பிரிப்பாற்றல்களில் 16 வண்ணங்களில் செயல்படும்.

மிகத்தரமான படங்களைப் பெற மேலும் பல அதிக பிரிப்பாற்றல்களில் செயல்படும் SVGA(Super VGA) வகைத் திரைகளும் கிடைக்சின்றன. இவற்றில் பல இட்சக்கணக்கான வண்ணங்களிலிருந்து ஒரு சமயத்தில் 16 அல்லது 256 போன்ற அளவில் வண்ணங்களைத் தேர்வு செய்து, அவற்றைக் கொண்டு வரையலாம். ஒரே சமயத்தில் இலட்சக்கணக்கான வண்ணங்களைப் பார்க்க முடியாது.

CGA, EGA, VGAஅட்டைகளுக்கு உகந்த ஒரு வண்ண, பலவண்ணத் திரைச் சாதனங்கள் கிடைக்கின்றன. ஒருவண்ணத் திரையில், பல வண்ணங்களுக்குப் பதிலாக, ஒளியின் அளவு மாறுபட்டுத் தெரியும்.

HGA (Hercules Graphics adaptor)என்பது 720×348பிரிப்பாற்றலில், ஒரே வண்ணத்தில் படங்கள் வரையத் தோன்றிய அட்டை. HGA, CGA, EGA அட்டைகள் வழக்கிழந்து வருகின்றன. இனி வரும் ஆண்டுகளில் VGA மற்றும்மேம்பட்ட திரைகளே கிடைக்கும்.

நினைவில் நிற்க

1. உள்/வெளி அட்டைகள் CGA, EGA, VGA, SVGA, HGA என்று பல விதங்களில் இருக்கின்றன.

2. திரையில் தோன்றும் புள்ளிகளின் எண்ணிக்கையைப் பிரிப்பாற்றல் குறிக்கிறது.

3. ஒரு சமயத்தில் திரையில் பார்க்கக்கூடிய வண்ணங்களின் எண்ணிக்கைக்கு ஏற்றவாறு, அந்த வண்ணங்களைக் குறிப்பிடத் தேவையான பிட்டுகளின் எண்ணிக்கை மாறுபடும்.

பயிற்சி வினாக்கள்

1. பிரிப்பாற்றல் என்றால் என்ன? CGA, EGA, VGA அட்டைகள் எந்தப் பிரிப்பாற்றல்களில் வேலை செய்யும்?
2. உயர் பிரிப்பாற்றல், நடுத்தரப் பிரிப்பாற்றல் என்றால் என்ன? அவற்றுக்கு நினைவகத்தில் எவ்வளவு இடம் தேவை? ஏன்?

7.3 திரை, வண்ணத் தேர்வு

பேசிக் மொழியைக் கொண்டு படங்கள் வரைவது எவ்வாறு என்பதைத் காணப் போகிறோம். இதற்கு நம்மிடம் CGA அட்டையும், அதற்கேற்ற வண்ணத்திரையும் இருப்பதாக வைத்துக் கொள்வோம். முதலில், நமக்குத் தேவையான பிரிப்பாற்றலுக்குத் திரையைத் தயார் செய்ய வேண்டும். அடுத்தது தேவையான வண்ணங்களையும் தேர்வு செய்து கொள்ள வேண்டும், பிறகே வரைய ஆரம்பிக்க வேண்டும். திரை மற்றும் வண்ணத் தேர்வுகளை இங்கு காண்போம்.

திரைத் தேர்வு

SCREEN 1 என்னும் வாக்கியம் நடுத்தரப் பிரிப்பாற்றல் உள்ள படத்திரையையும்,

SCREEN 2 என்பது உயர் பிரிப்பாற்றல் படத்திரையையும் உருவாக்கும்.

வண்ணத்தட்டு (Palette)

ஓவியர்கள் வண்ணங்களைக் கலக்க, சில சிறு குழிகள் உள்ள தட்டைப் பயன்படுத்துவார்கள். அதற்கு palette என்பது பெயர். ஒரு தட்டில் 4 குழிகள் இருப்பதாக வைத்துக் கொள்வோம். இதில் 4 வண்ணங்களை வைத்துப் பயன்படுத்தலாம். இவற்றில் உள்ள வண்ணங்களை 0, 1, 2, 3 என்று குறிப்பிடலாம். இதில் 0 என்பது எப்போதும் திரையின் **பின்னணி வண்ணத்தைக்** (background color) குறிக்கும். இதை நம் விருப்பம் போல் தேர்ந்தெடுத்துக் கொள்ளலாம். மற்ற மூன்று வண்ணங்கள் நிலையானவை. நாம் அவற்றை மாற்ற முடியாது. இம்மாதிரித் தட்டுகள் சாதாரணமாக இரண்டு இருக்கும். சுழி எண்ணுள்ள தட்டில் (1)பச்சை, (2) சிவப்பு, (3) பழுப்பு(brown)எனும் மூன்று வண்ணங்களும், ஒன்று என்ற எண்ணுள்ள தட்டில் (1) சியான் (cyan), (2)மஜந்தா (magenta), (3) வெள்ளை எனும் மூன்று நிறங்களும் நிலையானவை. நமக்குத் தேவையான தட்டையும் (0அல்லது 1), அதன் 0-வண்ணமான பின்னணி வண்ணத்தையும் தேர்ந்தெடுத்துக் கொள்ளலாம். பின்னணி வண்ணம் சாதாரணமாக, கீழே உள்ள 16 வண்ணங்களில் (அல்லது முதல் 8 வண்ணங்களில்) எது வேண்டுமானாலும் இருக்கலாம்.

0. கறுப்பு	4.சிவப்பு
1.நீலம்	5.மஜந்தா
2.பச்சை	6.பழுப்பு
3.சியான்	7.வெள்ளை

8.சாம்பல்	*12. இளஞ்சிவப்பு*
9.வெளிர்நீலம்	*13.வெளிர்மஜந்தா*
10.வெளிர்பச்சை	*14.மஞ்சள்*
11.வெளிர்சியான்	*15. அதிவெள்ளை*

COLOR *என்னும் வாக்கியம் பின்னணி வண்ணத்தையும், வண்ணத்தட்டையும் தேர்வு செய்யப் பயன்படும். இதன் இலக்கணம்* COLOR background color, palette number. *கவனிக்கவும்.* COLOUR *இல்லை,* COLOR. *உதாரணமாக,* COLOR 1, 0 *என்னும் வாக்கியம்* 0 *எண் தட்டுடன், பின்னணி வண்ணமாக நீலத்தைத் தேர்ந்தெடுக்கிறது. இப்போது* 0-*நீலம்,* 1-*பச்சை,* 2-*சிவப்பு,* 3-*பழுப்பு என்ற நிறங்களை திரையில் பயன்படுத்தலாம்.* COLOR 3, 1 *என்றால் 0-சியான், 1-சியான், 2-மஜந்தா, 3-வெள்ளை நிறங்கள் கிடைக்கும். இவற்றில் சியான் பின்னணி நிறம். பின்னணி நிறமானது தட்டில் நிலையாக உள்ள நிறங்களில் ஒன்றாகவும் இருக்கலாம்.*

வண்ணங்களைத் தேர்ந்தெடுப்பது நடுத்தரப் பிரிப்பாற்றல் திரைக்குத்தான் பொருந்தும். உயிர்பிரிப்பாற்றலுக்குப் பொருந்தாது.

படத்திரையில் இருக்கும்போது அதில் எழுத்துக்களை எழுதமுடியுமா? எழுத்துத் திரையில் இருப்பதாக நினைத்துக்கொண்டு, PRINT *வாக்கியம் மூலம் வழக்கம் போல் எழுதலாம்.*

நினைவில் நிற்க

1. SCREEN வாக்கியம் படத்திரையைத் தேர்வு செய்ய உதவும்.

2. நடுத்தரப் பிரிப்பாற்றல் திரைக்கு வண்ணங்களைத் தேர்ந்தெடுக்க COLOR வாக்கியம் பயன்படும். இதில் பின்னணி வண்ணம், தட்டு எண் என்ற இரண்டையும் கொடுக்க வேண்டும்.

பயிற்சி வினாக்கள்

1. நடுத்தரப் பிரிப்பாற்றல் திரைக்கு வண்ணங்களைத் தேர்ந்தெடுப்பது எவ்வாறு என விவரி.
2. நடுத்தரப் பிரிப்பாற்றல் திரையில் கீழ்க்காணும் 4 வண்ணங்கள் ஒரே சமயத்தில் திரையில் தோன்ற முடியுமா என்று கூறு. காரணமும் கூறு.

அ. சிவப்பு, மஞ்சள், பச்சை, பழுப்பு

ஆ. சிவப்பு, மஞ்சள், கறுப்பு, வெள்ளை

இ. சிவப்பு, மஜந்தா, கறுப்பு, பச்சை

ஈ. சியான், மஜந்தா, நீலம், கறுப்பு

3. வண்ணத் தட்டு என்றால் என்ன? அதன் பயன் யாது?

4. கீழே உள்ள வாக்கியங்களில் தவறுகள் எவையேனும் இருந்தால் கூறு.

அ. COLOR 5, 2

ஆ. COLOR 6, 1

இ. COLOR -1, -2

ஈ. COLOR 2, 800

உ. COLOR, 1

ஊ. COLOR 6

எ. COLOR 0, 0

7.4 புள்ளிகள்

படங்கள் பல புள்ளிகளால் ஆனவை. நேர்கோடு, செவ்வகம், வட்டம், நீள்வட்டம் போன்று சீராக வரும் புள்ளிகளை இட, கட்டளைகள் தனியாக உள்ளன. சீராக வராமல், சில புள்ளிகளை மட்டும் தனியாக இடுவது எவ்வாறு என்பதை முதலில் இப்பாடத்தில் பார்ப்போம்.

புள்ளிகளின் அமைப்பு

நடுத்தரப் பிரிப்பாற்றல் திரையில் வரிசைக்கு 320 புள்ளிகளும், அவ்வாறு 200 வரிசைகளும் உள்ளன. இந்தப் புள்ளிகள் ஒரு வரைபடக் காகிதத்தில் உள்ள கோடுகள் வெட்டிக் கொள்ளும் இடங்களைக் குறிப்பதாக நினைக்கலாம். வரைபடத்தில் x-அச்சும், y-அச்சும் வெட்டும் (0,0)என்ற புள்ளி பெரும்பாலும் காகிதத்தின் இடது கீழ் மூலையில் அல்லது காகிதத்தின் நடுவில் இருக்கும். ஆனால் திரையில் (0,0) என்ற புள்ளி இடது மேல் மூலையில் இருக்கும். x-அச்சு இடமிருந்து வலமாகவும், y-அச்சு மேலிருந்து கீழாகவும் இருக்கும். அதனால், வலதுபுற மேல் மூலை (319, 0) என்ற புள்ளியையும், வலதுபுறக் கீழ் மூலை (319, 199) என்பதையும், இடதுபுறக் கீழ் மூலை (0, 199)என்ற புள்ளியையும் குறிக்கும்.

உயர் பிரிப்பாற்றலில் இந்தப் புள்ளிகள் முறையே (639, 0), (639, 199), (0, 199) என இருக்கும்.

தொடக்கத்தில்

ஒரு படத்திரையை SCREEN வாக்கியம் மூலம் தெரிவு செய்தவுடன், அதிலுள்ள எல்லாப் புள்ளிகளும் கறுப்பு வண்ணத்தில் அமையும். நடுத்தரப் பிரிப்பாற்றலில், COLOR வாக்கியம் மூலம் வேறு பின்னணி வண்ணத்தைத் தேர்ந்தெடுத்தால் உடனே பின்னணி வண்ணம் மாறும்.

புள்ளிகள் வைக்க, நடுத்தரப் பிரிப்பாற்றலில் 0 முதல் 3 வரை 4 வண்ணங்களும், உயர் பிரிப்பாற்றலில் 0, 1 என இரண்டு வண்ணங்களும் உண்டு. இதில் எதையும் பயன் படுத்தலாம்.

நடுத்தரப் பிரிப்பாற்றல்

இதில் புள்ளியை வைக்க இரண்டு வாக்கியங்கள் உண்டு. அவை-

PSET (X,Y), COLOR

PRESET (X,Y), COLOR

Point set, Point reset என்பவற்றின் குறுக்கங்கள் இவை. நிற எண்ணைக் குறிப்பிட்டால் இவை இரண்டுமே ஒரே விதத்தில் செயல்பட்டு (x,y) என்ற புள்ளியை அந்த நிறத்தில் வைக்கும். இங்கு,

x என்பது 0 முதல் 319 வரையிலும்,

y என்பது 0 முதல் 199 வரையிலும்

color என்பது 0 முதல் 3 வரையிலும், இருக்க வேண்டும்.

உதாரணங்கள்

PSET (10, 140), 2 என்பதும் PRESET (10, 140), 2 என்பதும் சமம்.

வண்ணம் குறிப்பிடப்படாவிட்டால், PSETஎன்பது 3 என்பதை முன்னணி வண்ணமாக பாவித்து, அந்தப் புள்ளியை முன்னணி வண்ணத்தில் இடும். ஆனால், வண்ணம் குறிப்பிடப்படாதபோது, PRESET என்பது பின்னணி வண்ணமான சுழி எண் வண்ணத்தில் அந்தப் புள்ளியை இடும். இதுதான் இவற்றுக்கு இடையே உள்ள வேறுபாடு. உதாரணமாக,

PSET (205, 67) என்பது PSET (205, 67), 3 என்பதற்குச் சமம்

PRSET (200, 20) என்பது PSET (200, 20), 0 என்பதற்குச் சமம்.

கவனிக்கவும். வண்ணத்தைக் குறிப்பிட்டால், அதற்கு முன் ஒரு காற்புள்ளி அவசியம். ஒரு புள்ளியைப் பின்னணி வண்ணத்தில் இடுவது என்பது அதை அழிப்பதற்குச் சமம்.

இப்போது பார்த்தது நடுத்தரப் பிரிப்பாற்றல் திரைக்குத்தான்.

உயர் பிரிப்பாற்றல்

இங்கு x என்பது 0 முதல் 639 வரையிலும்,

y என்பது 0 முதல் 199 வரையிலும்,

color என்பது 0 அல்லது 1 என்றும் இருக்க வேண்டும்.

வண்ணத்தை 2 அல்லது 3 என்று கூடக் கொடுக்கலாம், அப்போது 2 என்பதை 0 எனவும், 3 என்பதை 1 எனவும் எடுத்துக் கொள்ளும். வேறு வித்தியாசமில்லை.

(x,y) என்னும் புள்ளி திரைக்குள் இருக்க வேண்டும் என்பதற்காக x, y இவற்றின் எல்லைகளைக் குறிப்பிட்டோம். எல்லைக்கு வெளியே போனால்? தவறு எதுவுமில்லை. ஆனால் அந்தப் புள்ளி திரையில் தோன்றாது. வேறு எதுவும் நடக்காது. உதாரணமாக, நடுத்தரப் பிரிப்பாற்றலில் PSET (20, 470), 2 என்பதும், உயர் பிரிப்பாற்றலில் PRESET (700, 20), 1 என்பதும் எதுவும் செய்யாது.

color என்பது 0 முதல் 3 வரைதான் இருக்க வேண்டும். வேறு எண்களாக இருந்தால் தவறு என்று கூறிவிடும்.

இப்போது சில படங்களை வரைந்து பார்ப்போம்.

1. $y = x^2 + 2$ *என்ற சமன்பாட்டிற்கான வரைபடத்தை வரை.*

இருவிதத் திரைகளிலும் y என்பது 0 முதல் 199 வரை இருக்கிறது. அதனால் x என்பது 0 முதல் 14 வரைதான் இருக்க முடியும். $14^2 = 196$, $15^2 = 225$.

```
10 SCREEN 2 :  REM HIGH RESOLUTION
20 FOR  X =  0 TO 14
30        Y  = X * X + 2
40        PSET(X,Y),  1
50 NEXT  X
60 END
```

2. நீலப் பின்னணியில் 100 சிவப்பு நட்சத்திரங்கள் மின்னுவதாகக் காண்பி.

ஒரு புள்ளியை மாற்றி மாற்றி வரைந்து அழித்தால் அது மின்னுவது போல் தோன்றும். நட்சத்திரம் சிவப்பு நிறத்தில் இருக்க வேண்டும் என்பதால் 0 எண் வண்ணத் தட்டையும், பின்னணிக்கு நீல வண்ணத்தையும் தேர்ந்தெடுக்க வேண்டும். நூறு புள்ளிகளின் இடங்களை முறையில்லாமல் தேர்ந்தெடுப்பது நல்லது. இங்கு நூறு xகளும் (0 முதல் 319 வரை), நூறு yகளும் (0 முதல் 199 வரை) தேவை. இவற்றை முதலில் கணித்து வைத்துக் கொள்ள வேண்டும். தேவையான போலிக்குறியீடு.

1. நூறு (x, y)களை முறையில்லாமல் தேர்ந்தெடு
2. நடுத்தரப் பிரிப்பாற்றல் திரையைத் தேர்ந்தெடு
3. வண்ணங்களைத் தேர்ந்தெடு
4. 1 முதல் 10,000 வரை செய் (பத்தாயிரம் முறை மினுக்கு)

   100 புள்ளிகளை சிவப்பில் வை.

   100 புள்ளிகளை நீலத்தில் வை.

```
10  DIM X  (100), Y (100)
20  RANDOMIZE
30  FOR I =1 TO 100
40          X (I) = INT (RND*320) :  REM GIVES  0 TO  319
50          Y (I) = INT (RND*200) :  REM GIVES  0 TO  199
60  NEXT I
70  REM
100 SCREEN 1
110 COLOR 1, 0 : REM BACKGROUND BLUE.THEN G, R, B
```

```
120 REM
130 FOR I = 1 TO 10000
140     FOR J  =  1 TO 100
150             PSET (x  (I), Y (I)), 2
160     NEXT J
170 REM SHOWING IN RED OVER
180 REM NOW ERASE
190     FOR J  = 1 TO 100
200             PSET (x (I), Y (I)), O
210     NEXT J
220  NEXT I
230  END
```

3. கறுப்பு வண்ணத்திரையில், பச்சை வண்ணத்தில் ஒரு செவ்வகம் வரை. அதற்கு உள்ளேயும் பச்சை நிறம் இருக்க வேண்டும். (15, 20), (25, 20), (25, 40), (15, 40) என்பவை செவ்வகத்தின் மூலைகள்.

```
10  SCREEN 1
20  COLOR 0, 0 :  REM BLACK BACKGROUND  0  PALETTE
30  REM
40  FOR X = 15 TO 25
50      FOR Y = 20 TO 40
60              PSET (X,Y), 1: REM 1  IS  GREEN
70      NEXT Y
80      NEXT X
90  END
```

நினைவில் நிற்க

1. திரையில் (0,0) இடது மேல் புறம் இருக்கும். X திசை இடமிருந்து வலமும், Y திசை மேலிருந்து கீழும் இருக்கும்.

2. PSET, PRESET என்பவை புள்ளி வைக்கப் பயன்படும். இவற்றில் நிறத்தைக் கூறாவிட்டால், PSET முன்னணி வண்ணத்திலும், PRESET பின்னணி வண்ணத்திலும் புள்ளியிடும்.

3. திரைக்கு வெளியே உள்ள புள்ளிக்கு வண்ணமிட்டால், எதுவும் நடக்காது.

பயிற்சி வினாக்கள்

1. கீழே உள்ள மூன்று சமன்பாடுகளுக்கும் ஒரே திரையில் படங்கள் வரை ஒவ்வொன்றையும் ஒவ்வொரு வண்ணத்தில் வரை.

   $y = 2x + 17$; $y = x^2 + x + 7$; $y = \sqrt{x} + ABS(x)$

2. PSET க்கும் PRESET க்கும் உள்ள ஒற்றுமை வேற்றுமைகள் என்ன?

3. (100,50) என்ற புள்ளியை மையமாகவும், 40 புள்ளி அளவு பக்கத்தைக் கொண்டதாகவும் ஒரு சதுரம் வரை. இதில் எதிரெதிர்ப் பக்கங்கள் ஒரு நிறத்திலும், அடுத்தடுத்தபக்கங்கள் வேறு நிறத்திலும் இருக்க வேண்டும்.

7.5 நேர்கோடும் செவ்வகமும்

நேர்கோடு

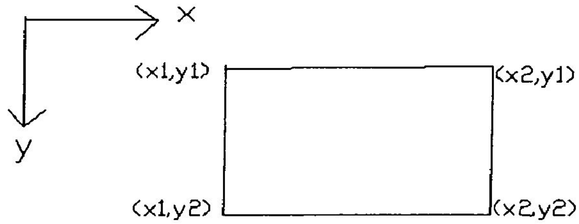
ஒரு நேர்கோட்டினை வரைய, அவற்றின் இரண்டு முனைகள் தேவை. எந்த நிறத்தில் வரைய வேண்டும் என்று கூற வேண்டும். LINE (x1, y1) - (x2, y2), color என்ற வாக்கியம் (x1, y1), (x2, y2) என்ற புள்ளிகளை இணைத்து color என்ற வண்ணத்தில் ஒரு நேர்கோடு வரையும். இங்கு x, y, color என்பவை நாம் சென்ற பாடத்தில் பார்த்ததுபோல் இருக்க வேண்டும். உதாரணம்-

LINE (10, 40) - (5, 150), 3

இதில் *(10, 40)* **ஆரம்பப் புள்ளி** எனப்படும். *(5, 150)* **முடிக்கும் புள்ளி** எனப்படும்.

செவ்வகம்

சாதாரணமாக ஒரு செவ்வகத்தை நான்கு நேர்கோடுகளால் வரைய வேண்டும். ஆனால், செவ்வகத்தின் பக்கங்கள் x மற்றும் y திசைகளுக்கு இணையாக இருந்தால், செவ்வகத்தை எளிதாக ஒரே வாக்கியத்தின் மூலம் வரைந்து விடலாம். இவ்வாறு வரைவதற்கு , இரண்டு எதிரெதிர் மூலைகளைக் கொடுத்தால் போதும். ஏன்? இந்த மூலைகள் (x1, y1), (x2, y2) என்று இருந்தால், அடுத்த இரண்டு மூலைகள் (x1, y2), (x2, y1) என்றுதான் இருக்கும். அதனால் தேவையான நேர்கோடுகள் (x1, y1) முதல் (x1, y2) வரை, (x1, y2) முதல் (x2, y2) வரை, (x2, y2) முதல் (x2,y1) வரை, (x2, y1)முதல் (x1, y1) வரை. இவற்றைக் கணிப்பொறி வரைந்துவிடும். படம் 7.1 ஐப் பார்க்கவும்.



படம் 7 - 1 செவ்வகத்தின் மூலைகள்

LINE என்னும் வாக்கியம் செவ்வகத்தையும் வரையும். தேவையானது செவ்வகம், நேர்கோடு இல்லை என்று குறிப்பிட்டு ச் சொல்ல வேண்டும். இதற்கு வாக்கியத்தில், B (box) என்ற குறியீட்டைச் சேர்க்க வேண்டும் (காற்புள்ளியும் அடுத்து Bயும்). அதாவது, LINE (x1, y1) - (x2, y2), color, B

உதாரணம் - LINE (10, 20) - (100, 50), 2, B

இந்த வாக்கியம் செவ்வகத்தின் சுற்றுக்கோடுகளை மட்டும் வரையும். செவ்வகத்தின் உட்புறத்தையும், அதே நிறத்தினால் *(கவனிக்கவும்-வேறு நிறத்தினால் அல்ல)* நிரப்ப வேண்டுமானால் B என்பதற்கு பதிலாக BF (Box fill) என்று எழுத வேண்டும். உதாரணமாக,

```
10 SCREEN 1: COLOR 0, 0
20 LINE (15, 20) - (25, 40), 1, BF
```

என்பது கறுப்பு வண்ணத்திரையில் , பச்சை வண்ணத்தால் நிரப்பப்பட்ட ஒரு செவ்வகத்தை வரையும். இதையே PSETகொண்டு, இரண்டு FORவளையங்களைக் கொண்டு சென்ற பாடத்தில் வரைந்திருக்கிறோம். இதையே பல நேர்கோடுகளாலும் உருவாக்கலாம் - இப்படி.

```
FOR X = 15 TO 25
   LINE (X, 20) - (X, 40), 1
NEXT X
```

எல்லாத் தகவல்களும் தேவையா?

LINE (X1, Y1) - (X2, Y2), color, B *(அல்லது)* BF

என்பதில் உள்ள தகவல்களில் **- (X2, Y2)** என்பது கண்டிப்பாகக் கொடுக்கப்பட வேண்டும். மற்றவற்றைத் தேவையானால் கொடுக்கலாம். இல்லையென்றால் விட்டுவிடலாம். அப்படி விட்டுவிட்டால் இந்த வாக்கியம் எவ்வாறு செயல்படும் என்று ஒவ்வொன்றாகப் பார்ப்போம்.

1. Bஅல்லது BFஐ விட்டுவிட்டால் செவ்வகம் வரையாமல் கோடுவரையும். Bஐ விட்டுவிடும்போது அதற்கு முன்னால் உள்ள காற்புள்ளியையும் விட்டுவிட வேண்டும். இது போன்ற வாக்கியங்கள் எப்போதும் காற்புள்ளியில் முடியாது என்பதைக் கவனத்தில் கொள்வது அவசியம்.

2. color என்பதைக் கூறாமல் விட்டால், தானாக முன்னணி வண்ணத்தில் வரையும். *(இந்த வண்ணம், உயர் பிரிப்பாற்றலில் 1, நடுத்தரப் பிரிப்பாற்றலில் 3)*

color மற்றும் B/BF என்று இரண்டையும் விட்டுவிடும்போது காற்புள்ளி எதுவும் வரக்கூடாது, உதாரணம் -

LINE (20, 30) - (40, 50)

ஆனால் color வராமல் B/BF வந்தால், Bக்கு முன்னால் வரும் எல்லாக் *(இங்கு 2)* காற்புள்ளிகளும் அவசியம். உதாரணம் -

LINE (20, 30) - (40, 50), B

அப்போதுதான் கணிப்பொறிக்கு (நமக்கும் கூட) எந்தத் தகவலை விட்டு விட்டோம், எதைக் கொடுத்திருக்கிறோம் என்று தெளிவாகத் தெரியும். மேலே உள்ள உதாரணத்தில் இரண்டு காற்புள்ளிகளுக்கு நடுவில் ஒரு தகவல் விடப்பட்டுள்ளது என்பது பார்த்தவுடன் தெரிகிறது.

3. இந்த வாக்கியத்தில் (X1,Y1) *என்பதைக் கூட விட்டு விடலாம். உதாரணமாக,* LINE - (40, 70) *என்று எழுதுவது சரியே! இங்கு* - **குறி** *கண்டிப்பாக வர வேண்டும். நேர்கோடு எங்கு முடிய வேண்டும் என்பது கூறப்பட்டுள்ளது. ஆனால் எங்கிருந்து ஆரம்பிக்க வேண்டும் என்பது நேரடியாகக் கூறப்படவில்லை. இதை, ''தற்போதைய இடத்திலிருந்து ஆரம்பித்து'' என்று பொருள் கொள்ள வேண்டும்.* **தற்போதைய இடம்** (current position) *என்றால் என்ன?*

அ. படத்திரையை உருவாக்கும்போது இது திரையின் நடுப்புள்ளி ஒன்றினைக் குறிக்கும். அதாவது நடுத்தரப் பிரிப்பில் (160, 100); உயர் பிரிப்பில் (320, 100),

ஆ. ஒரு புள்ளி வைத்தால், அந்தப் புள்ளி,

இ. ஒரு கோடு வரைந்தால், அதன் கடைசிப்புள்ளி (X2, Y2)

ஈ. ஒரு செவ்வகம் வரைந்தால், அதன் கடைசிப்புள்ளி (இது அதன் முதற் புள்ளிக்குச் சமம்).

பொதுவாக, கடைசியாக நாம் வைத்த புள்ளிதான் தற்போதைய புள்ளி எனப்படும்.

```
LINE - (10, 20) - (10, 50), 2
LINE - (100, 50), 2
LINE - (100, 20), 2
LINE - (10, 20), 2
```

என்பது LINE (10, 20) - (100, 50), 2, B *என்பதற்குச் சமம்.*

சில படங்களை வரைந்து பார்ப்போம்.

1. ஒரு பச்சை நிறச் சதுரத்தின் அகலம் 180 புள்ளிகள். இதன் இடது மேற்புற மூலை (10, 10). இதன் நடுவில் 160 புள்ளி அகலம் உள்ள ஒரு சிவப்புச் சதுரமும், அதன் நடுவில் 140 புள்ளி அகலம் உள்ள ஒரு பழுப்புச் சதுரமும் வரை. சதுரங்களின் உட்புறமும், சுற்றுக்கோடும் ஒரே நிறமாக இருக்க வேண்டும்.

```
10 SCREEN 1
20 COLOR 0, 0: REM 1 = GREEN. 2 = RED .3 = BROWN
30 LINE (10, 10) - (190, 190), 1, BF
40 LINE (20, 20) - (180, 180), 2, BF
50 LINE (30, 30) - (170, 170), 3, BF
60 END
```

இதில் உள்ள மூன்று LINE வாக்கியங்களை எதிர்வரிசையில் எழுதினால் என்ன ஆகும்? அதாவது 30, 40, 50 வரி எண்களை 50, 40, 30 என மாற்றினால் என்ன படம் கிடைக்கும்?

2. திரையில் சில செய்திகளை அல்லது தகவல்களைக் காண்பிக்கும்போது, அவை ஒரு வண்ணச் செவ்வகத்திலும், அந்த வண்ணச் செவ்வகத்தின் நிழல் சற்றுத் தள்ளி, வேறொரு நிறத்திலும் காண்பிக்கப்படும். இந்தச் செவ்வகங்களை எவ்வாறு வரைவது? முதலில் நிழலையும், பிறகு அதன் மேல் இன்னொரு செவ்வகத்தையும் வரையலாம். நிழல் பழுப்பு நிறத்திலும், மேல் செவ்வகம் சிவப்பிலும் இருக்கட்டும். மேல் செவ்வகத்தின் இடதுமேற்புற முனை(50, 50) எனவும், செவ்வகத்தின் அகலம் 200 புள்ளிகளாகவும், உயரம் 100 புள்ளிகளாகவும் இருக்கட்டும். நிழல் 10 புள்ளிகள் வலது புறமும், 10 புள்ளிகள் கீழேயும் தெரிவதாக வைத்துக்கொள்வோம். இதற்கான கட்டளை நிரல்-

```
10 SCREEN 1
20 COLOR 0, 0 : REM 1 = GREEN 2 = RED 3 = BROWN
30 LINE (60, 60) - (260, 160), 3, BF
40 LINE (50, 50) - (250, 150), 2, BF
50 END
```

நினைவில் நிற்க

1. LINE என்ற வாக்கியமானது, நேர்கோடு, செவ்வகத்தின் சுற்றுக்கோடுகள், நிரப்பப்பட்ட செவ்வகம் என மூன்றையும் வரையும்.

2. நடுவில் வரும் சில தகவல்களை வாக்கியத்தில் கொடுக்காமல் விடும் போது, தேவையான காற்புள்ளி போன்றவற்றைக் கட்டாயம் இட வேண்டும்.

3. வாக்கியத்தைக் காற்புள்ளியில் முடிக்கக் கூடாது. கடைசியில் வரும் காற்புள்ளிகளை விட்டுவிட வேண்டும்.

4. எந்த நேரத்திலும், கடைசியாகப் புள்ளிவைத்த இடம் தற்போதைய இடம் எனப்படும்.

பயிற்சி வினாக்கள்

1. ஒரு செவ்வகத்தை வரைவதற்கான கட்டளை எது? அதில் என்னென்ன தகவல்கள் தரப்படவேண்டும்?
2. கீழே உள்ள கட்டளைகள் வரையும் படங்களை விவரி. வரைந்து காண்பி.

அ.
```
10 SCREEN 1
20 COLOR 6,1
30 LINE (40, 30 ) - (70, 80), 3
```

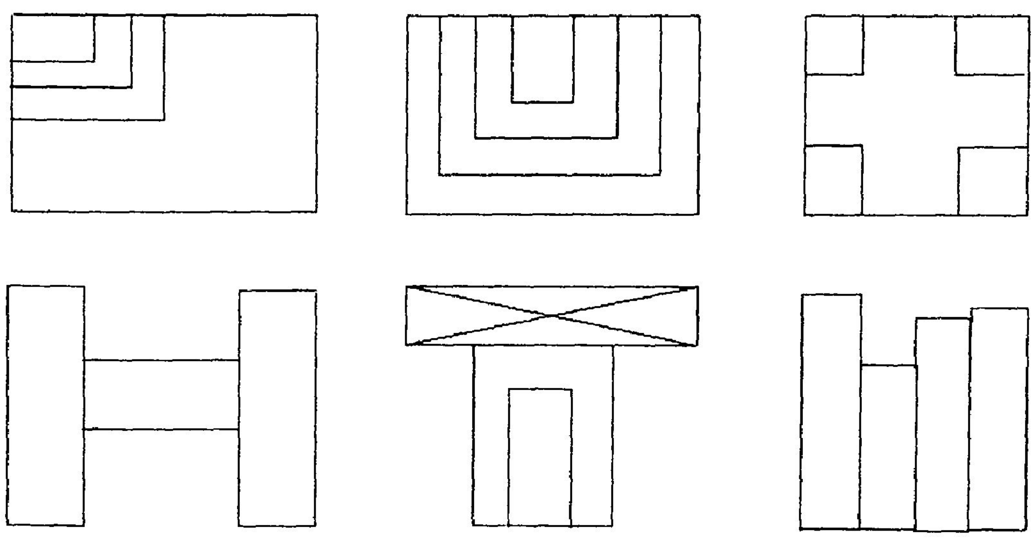
```
40 LINE - (100, 120), 2, BF
50 LINE - (120, 120), 1, B
60 LINE - (20, 10) - (40, 30)
70 LINE - (50, 50), B
```

ஆ. 10 SCREEN 1

```
20 COLOR 2, 0
30 RANDOMIZE
40 FOR I = 1 TO 100
50        LINE - (INT (RND* 200), INT (RND* 100)), INT (RND* 4)
60 NEXT I
70 END
```

இ. மேலே உள்ள கட்டளை நிரலில் 50 ஆவது வரியை இப்படி மாற்று.

```
50. LINE (160, 100) - (INT(RND * 320), INT (RND* 100)), INT (RND* 4)
```



படம் 7 - 2 நேர் கோட்டுப் படங்கள்

3. கீழே உள்ள வரிகளில் தவறுகள் எவையேனும் இருந்தால் கூறு.

அ. LINE (10, 20) - 3, B

ஆ. LINE (10, 10), (100, 100), 20, BF

இ. LINE - (7, 6)

ஈ. LINE - (7, 6), F

உ. LINE (7, 6) - (A, B), B

ஊ. LINE (6, 7) - (B, A) B, 2

4. படம் 7.2. இல் காணப்படும் படங்களை வரைய, கட்டளை நிரல்கள் எழுதுக. முதலில் புள்ளிகளின் இடங்களை, தேவைக்கேற்றபடி வைத்துக் கொள்.

7.6 தொடர்ந்து வரைவது

ஒரு படத்தில் பல நேர்கோடுகள் இருந்தால் அவற்றை வரைய ஓர் எளிய முறை உள்ளது. பல கோடுகளை ஒரே கட்டளை மூலம் வரைந்துவிடலாம். ''இங்கிருந்து கிழக்கில் 100 மீட்டர் போ. அங்கிருந்து வடக்கில் 50 மீட்டர், பிறகு மேற்கே 100 மீட்டர், அடுத்து தெற்கே 50 மீட்டர் போ. கடைசியில் எங்கே இருப்பாய்?'' - இந்தக் கேள்விக்கு உங்களிடமிருந்து உடனே விடை கிடைத்திருக்கும், ''புறப்பட்ட இடத்திலேயே இருப்பேன்'' என்று. ''நீ சென்ற பாதை எப்படியிருக்கும்?'' என்பதற்கு ''அது ஒரு செவ்வகம்'' என்றும் கூறுவீர்கள்.

கணிப்பொறி கோடுகள் வரைய இதே போல் உத்தரவிடலாம். இதற்கு மொத்தம் எட்டுத்திசைகளைப் பயன்படுத்தலாம். அவை U (up), D(down), L(left), R(right), E(north-east), F(south-east), G(south-west), H(north-west). வரிசையாக, எந்தத்திசையில் எவ்வளவு புள்ளிகள் செல்லவேண்டும் என்பதைக் கூறவேண்டும். DRAW என்பது இதற்கான வாக்கியம். DRAW "R 100 U 50 L 100 D 50" வாக்கியம் நாம் மேலே சொன்ன செவ்வகத்தை வரையும், மீட்டர் என்பதைப் புள்ளி என்று மாற்றினால். இந்த வாக்கியத்தில் இரண்டினைக் கவனிக்க வேண்டும்.

1. கொடுக்கப்படும் செய்தி மேற்கோள் குறிக்குள் இருக்கிறது. அதனால் இது ஒரு சர மாறிலி.

2. நாம் படிப்பதற்கு வசதியாக, தேவையான இடைவெளிகளை விட்டு எழுதலாம். இடைவெளிகள் இங்கு எதையும் குறிக்காது.

இந்த வாக்கியத்தில் மேலும் சில தகவல்களைச் சேர்த்து வாக்கியத்தின் பயன்பாட்டை விரிவாக்கலாம். அவற்றில் எளிமையான சிலவற்றை மட்டும் இங்கு பார்ப்போம். திசைகளைக் கொடுத்து வழி நடத்திக் கொண்டிருக்கும் போது, நடுவில், ஒரு குறிப்பிட்ட புள்ளிக்குப் போ (உதாரணமாக (20, 50) க்குப் போ) என்று கூற வேண்டி வரலாம். M(move) என்ற எழுத்து இதைக் குறிப்பிடும்.

DRAW "L 25 M 20, 50 F 5"

என்றால், இப்போதைய இடத்திலிருந்து இடதுபக்கம் 25 புள்ளி செல். அங்கிருந்து (20, 50)க்குப் போ. அங்கிருந்து தென்கிழக்கே 5 புள்ளி செல் என்று பொருள். இங்கு M x,y என்று வருவதில் x இன் முன் +, - என்ற எந்தக் குறியீடும் இருக்கக் கூடாது. அப்படி இருந்தால் பொருள் மாறிவிடும். + அல்லது -குறி இருந்தால் இந்த (x,y) என்ற புள்ளியை வேறுவிதத்தில் கணக்கிடவேண்டும். புதிய புள்ளி இப்போதைய இடத்தைச் சார்ந்ததாகிவிடும். இப்போதைய இடத்தை (0, 0)என்று வைத்துக் கொண்டு பார்த்தால் (x, y) எங்கிருக்கும் என்று பார்த்து, அந்தப் புள்ளியை எடுத்துக் கொள்ள வேண்டும். உதாரணமாக, தற்போதைய இடம் (120, 50) என்றால், M + 10, + 5 என்பது (120 + 10, 50 + 5) க்குப் போ என்பதற்குச் சமம்.

கணிப்பொறியின் கையில் இருக்கும் பேனா ஓர் இடத்திலிருந்து இன்னொரு இடத்திற்குச் செல்லும்போது கோடு வரைந்து கொண்டே செல்லும். கோடு வரையாமல் செல்ல வேண்டும் என்றால், அந்தக் கோட்டுக்கான குறியீட்டுக்கு முன்னால் B(for blank)என்னும் எழுத்தைச் சேர்க்க வேண்டும். உதாரணமாக, DRAW "U10 BR20 D10 L20" என்பது ப என்ற எழுத்தை உருவாக்கும். DRAW "U10 BR20 D10 BL20"என்பது || என்ற வடிவத்தை உருவாக்கிவிட்டுப் புறப்பட்ட இடத்திற்கே வந்து சேரும்.

கோடுகளை நமக்குத் தேவையான வண்ணங்களில் வரையலாம். Cn என்றால், வரையும் வண்ணத்தை n என்ற நிறமாக வைத்துக் கொள்ளும். DRAW "C1 U10 C3 E8 R25 C2 L20"என்பதில் உள்ள நான்கு கோடுகளும் முறையே *1, 3, 3, 2* என்ற வண்ணங்களில் வரையப்படும்.

நினைவில் நிற்க

1.பல நேர்கோடுகளை வரிசையாக வரைய DRAW பயன்படும். இதில் U, D, L, R, E, F, G, H, M, B, C என்பவற்றைப் பயன்படுத்தலாம்.

2. M, X, Y என்பதில், X இன் முன்னால் + அல்லது - குறி சேர்த்தால், புதிய புள்ளி, தற்போது இருக்கும் இடத்தைச் சார்ந்து கணிக்கப்படும்.

பயிற்சி வினாக்கள்

1. DRAW வாக்கியத்தின் பயன்பாட்டை விவரி.

2 DRAW வாக்கியத்தில் பயன்படும் M, B குறியீடுகளின் செயல்பாட்டை விவரி.

3. கீழ்க்காணும் வாக்கியங்கள் வரையும் படங்களை விவரி. வரைந்து காண்பி.

அ.DRAW "BM 10, 20, D 50 R 40 BM 100, 100, E20 F20"

ஆ.DRAW "BM 100, 10 F 50 BH 40 M100, 20M + 20, 20"

இ. DRAW "C2 BM 10, 20 G 5 F50 C1 D10"

ஈ. DRAW "BM 100, 50 C0 R 5 C1 U5 C2 M - 5, 5"

ஊ.DRAW "M50, 50 M -10, 5B M 20, 5M10, 10"

4. கீழ்க்காணும் வாக்கியங்களில் தவறுகள் எவையேனும் இருந்தால் கண்டுபிடி.

அ.DRAW M10,5,D50 K40

ஆ.DRAW C50 D50 E50

இ. DRAW "C2 C0 C1 D50 F40"

ஈ. DRAW "C0 M 20, 20M - 20, 50 BM7 , 5"

உ. DRAW "U5 U7 U2"

ஊ.DRAW "CM2, 5R 5, 5"

எ. DRAW "L22, U5, 55"

ஏ. DRAW "L5 R8 D6", "U5 F7"

7.7 வட்டம்

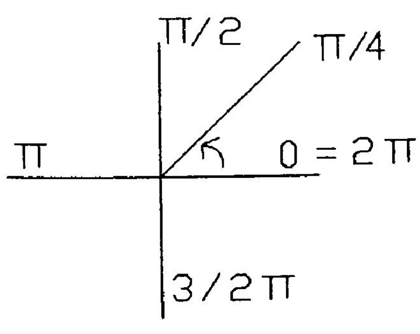
வட்டம், வட்டத்தின் ஒரு பகுதி, நீள் வட்டம் (ellipse), நீள் வட்டத்தின் ஒரு பகுதி - இவை எல்லாவற்றையும் CIRCLE என்ற ஒரு வாக்கியத்தைக் கொண்டே வரையலாம். இதன் பொது இலக்கணம் CIRCLE (x, y) radius, color, start, angle, end angle, aspect, ratio. இதில் x, y, radius என்ற மூன்றும் அவசியம். மற்றவற்றைத் தேவையென்றால் கொடுக்கலாம். கொடுக்காதபோது கணிப்பொறி தானாக இவற்றை முடிவு செய்து கொள்ளும். வாக்கியத்தைக் காற்புள்ளியுடன் முடிக்கக் கூடாது; நடுவிலுள்ள தகவல்களை விட்டுவிடும்போது காற்புள்ளிகள் சரியாக இடப்பட வேண்டும் என்ற பொதுவிதிகள் நினைவிருக்கட்டும்.

முழுவட்டம்

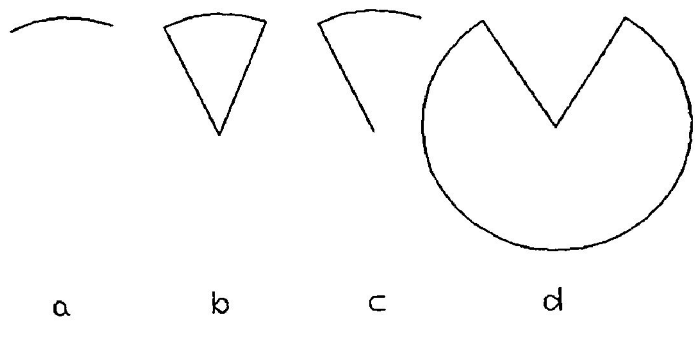
முதலில் முழுவட்டம் வரைவதுடன் தொடங்கலாம். வட்டம் வரைய அதன் மையப்புள்ளியும் (x,y) ஆரமும் (r) தேவை. அதுவே போதும் . CIRCLE (50, 100), 20 என்பது (50, 100) ஐ மையமாகவும், 20 புள்ளிகளை ஆரமாகவும் கொண்ட ஒரு வட்டம் வரையும். நிறம் கொடுக்கப்படாததால் முன்னணி வண்ணத்தில் வரையும்.

CIRCLE (100, 100), 50, 2

என்பது இரண்டாம் வண்ணத்தில் , 50 புள்ளி ஆரமுள்ள வட்டம் வரையும்.



படம் 7 - 3 கோணங்கள்



படம் 7 - 4 வட்டத்தின் பகுதிகள்

வட்டத்தின் பகுதி

வட்டத்தின் ஒரு பகுதியை மட்டும் வரைய, அந்தப் பகுதியின் ஆரம்ப கோணத்தையும், முடிவு கோணத்தையும் கொடுக்க வேண்டும். இந்தக் கோணங்களை, வலது பக்க கிடைக்கோட்டிலிருந்து, கடிகார ஓட்டத்திற்கு எதிர்ப்புறம் நகர்ந்து கணக்கிட வேண்டும். கோணங்களை ரேடியனில்தான் கொடுக்க வேண்டும். சில முக்கியக் கோணங்களை படம் 7.3 இல் காணலாம்.

படம் 7.4.a இல் உள்ளதுபோல் ஒரு கால்வட்டத்தை வரைவது எப்படி? இங்கு ஆரம்ப கோணம் 45^0, முடிவு கோணம் 135^0. இவற்றை ரேடியனில் $\pi/4$, $3\pi/4$ என்று குறிப்பிட வேண்டும். π என்ற குறியீடு பேசிக் மொழியில் இல்லை. அதனால் அதை PI என்ற மாறியாக்கி, π இன் தோராயமான மதிப்பான 3.141593 என்பதைக் கொடுப்போம். இப்போது கீழே உள்ள கட்டளைகள் தேவையான கால் வட்டத்தைக் கொடுக்கும்.

```
10 SCREEN 1
20 COLOR 0, 1
30 PI = 3.141593
40 CIRCLE (100, 50), 20, 1, PI/4, 3* PI/4
```

படம் 7.4.b இல் வட்டப் பகுதியின் இரு முனைகளும், மையப்புள்ளியுடன் சேர்க்கப்பட்டுள்ளன. படம் 7.4.c இல் ஒரு முனை மட்டும் அவ்வாறு சேர்க்கப்பட்டுள்ளது. ஒரு முனையிலிருந்து மையப்புள்ளிக்கு கோடு வேண்டுமானால் அந்த முனையைக் குறிக்கும் கோணத்திற்கு முன்னால் ஒரு கழித்தல் குறி சேர்க்க வேண்டும். அவ்வளவுதான். சாதாரணமாக எந்தக் கோணமும் 0 முதல் 2π வரை நேர்மறை எண்ணாகத்தான் கொடுக்கப்படும். எதிர்மறை எண்ணாக கொடுக்கப்படாது. அதனால் கழித்தல் குறி இருக்கும்போது, அது ஒரு ஆரத்தை வரைய என்பது தெளிவாகத் தெரியும். கொடுத்த படங்களை (படம் 7.4.b,c) வரையத் தேவையான வாக்கியங்கள்-

```
50 CIRCLE (100, 50), 20, 1, - PI/4, - 3* PI/4
60 CIRCLE (100, 50), 20, 1, PI/4, - 3* PI/4
```

படம் 7.4.d இல் உள்ள முக்கால் வட்டத்தை எப்படி வரைவது? இங்கும் கோணங்கள் $\pi/4$ மற்றும் $3\pi/4$ என்றுதானே உள்ளன? கவனித்துப்பாருங்கள். கோணத்தைக் கணக்கிடும்போது எப்போதும் கடிகார ஓட்டத்திற்கு எதிர்ப்புறமாக நகர்வதால், ஆரம்ப கோணம் $3\pi/4$, முடிவு கோணம் $\pi/4$. முடிவு கோணம் ஆரம்ப கோணத்தைக் காட்டிலும் குறைவாக இருக்கும்போது இம்மாதிரிப் பகுதி கிடைக்கும். தேவையான கட்டளை-

```
CIRCLE (100, 50), 20, 2, - 3* PI/4, - PI/4
```

ஆரம்ப கோணமும், முடிவு கோணமும் கொடுக்கப்படாதபோது அவை முறையே 0,2π என எடுத்துக் கொள்ளப்படும். அப்போது ஒரு முழு வட்டம் கிடைக்கும். அதனால்தான் முழு வட்டத்தை வரையும் கட்டளைகளில் இவற்றை விட்டு விடுகிறோம்.

அகல-உயர விகிதம் (aspect ratio)

நீள்வட்டங்களைப் பற்றித் தெரிந்து கொள்வதற்கு முன்னால், திரையைப் பற்றி மேலும் சிறிது தெரிந்துகொள்வது அவசியம். **அகல-உயர விகிதம்** என்பது பற்றி இப்போது பார்ப்போம். சாதாரணமாக திரையின் அகலமும், உயரமும் 4:3 என்ற விகிதத்தில் இருக்கும். அதனால் திரையின் அகலம் 4 அலகுகள், உயரம் 3 அலகுகள் என வைத்துக் கொள்ளலாம். நடுத்தரப் பிரிப்பாற்றலில் வரிசைக்கு 320 புள்ளிகள் இருக்கும். அதனால் ஒரு புள்ளியின் அகலம் 4/320 அலகுகள். மொத்தம் 200 வரிசைப் புள்ளிகள் இருப்பதால், ஒரு புள்ளியின் உயரம் 3/200 அலகுகள். அதாவது ஒரு புள்ளியின் அகலம் 4/320 = 1/80 = 5/400 அலகுகள். உயரம் 6/400 அலகுகள். ஒவ்வொரு புள்ளிக்கும் அகலத்தைவிட உயரம் சற்று அதிகமாக இருக்கிறது. இதை, நடுத்தர பிரிப்பாற்றல் திரையின் அகல-உயர விகிதம் 5/6 என்று குறிப்பிடலாம். இதே போல் உயர் பிரிப்பாற்றல் திரையின் அகல-உயர விகிதம் 5/12.

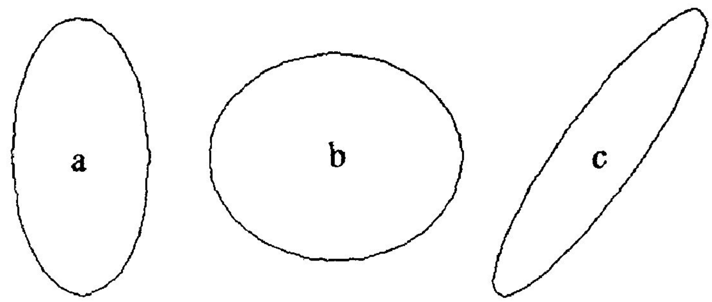
வட்டமா, நீள் வட்டமா?

(50, 50) என்ற புள்ளியை மையமாகவும், 10 புள்ளி ஆரமும் உள்ள ஒரு வட்டத்தை எவ்வாறு வரைவது என்று பார்ப்போம். இந்த வட்டத்தில் உள்ள புள்ளிகள் எல்லாம் $(x - 50)^2 + (y - 50)^2 = 10^2$ என்ற சமன்பாட்டின் தீர்வுகளாக அமைய வேண்டும். பின்னங்கள் வரும்போது அவற்றை முழு எண்களாக மாற்றிக் கொள்வோம். முதலில் எளிதாகக் கணிக்கக் கூடிய 4 புள்ளிகளைப் பார்ப்போம். மையப்புள்ளிக்கு மேலும், கீழும், இடமும், வலமும் 10 புள்ளிதூரத்தில் உள்ள புள்ளிகள் தான் இவை ; முறையே (50, 40),(50, 60),(40, 50),(60, 50). இவற்றுக்கும் மையப்புள்ளிக்கும் இடையே உள்ள உண்மையான தூரங்கள் என்ன?

இடவலமாக அது 10 X 5/400 = 5/40 அலகுகள்.

மேல் கீழாக அது 10 X 6/400 = 6/40 அலகுகள்

அதாவது வட்டத்தின் சமன்பாட்டை வைத்து நாம் வரையும்படம் சரியான வட்டமாகத் தோற்றமளிக்கப் போவதில்லை. அகலத்தைவிட உயரம் சற்று அதிகமாக, கிட்டத்தட்ட முட்டை வடிவத்தில், நீள் வட்டமாகத்தான் (ellipse)காட்சியளிக்கும்! (படம் 7.5.a)



படம் 7 - 5 நீள் வட்டங்கள்

இதை எப்படிச் சரி செய்து வட்டமாகக் காட்சி தரச் செய்வது? இதற்கு உயரத்தைச் சற்றுக் குறைக்க வேண்டும். அதாவது (மையப்புள்ளி வழியே செல்லும்) கிடையாக உள்ள விட்டத்திற்கு மேலும் கீழும் உள்ள உயரத்தை 5/6 என்னும் பின்னத்தால் பெருக்க வேண்டும். இவ்வாறு செய்தால் உயரம் குறைந்து வட்டமாகக் காட்சி தரும் (6/40) X (5/6) = 5/40 அலகுகள்).

உயரத்தை இவ்வாறு குறைப்பது பற்றி நாம் எதுவும் குறிப்பிடவில்லையென்றால், தகவல் கொடுக்கவில்லையென்றால், கணிப்பொறி தானாகவே, தேவையான பின்னத்தைக் கணக்கிட்டு, உயரத்தை மாற்றி வட்டமாக வரைந்து காட்டும். இந்த பின்னம், நடுத்தரப் பிரிப்பாற்றலுக்கு 5/6, உயர் பிரிப்பாற்றலுக்கு 5/12. இவை வேறொன்றுமில்லை. அகல-உயர விகிதங்கள்தான்.

நீள் வட்டங்கள்

உயரத்தை 5/6 என்ற அளவில் குறைக்காமல், 2/6 என்ற அளவில் குறைத்தால் என்ன ஆகும்? உயரம் குறைந்து குட்டையான நீள் வட்டமாகக் காட்சியளிக்கும் படம் 7.5-b இல் உள்ளதுபோல். நீள் வட்டங்களை வரைய ஒரு வழி கிடைத்துவிட்டது! இந்த அகல - உயர விகிதத்தையும் நாமே கொடுப்பதுதான் அது . CIRCLE வாக்கியத்தின் கடைசித் தகவல் இந்த பின்னம்தான். இது கொடுக்கப்படாதபோது கணிப்பொறி இதைத் தானாக, திரையின் அகல - உயர விகிதமாக எடுத்துக் கொண்டு, வட்டங்களை வட்டங்களாகவே வரையும்.

ஒரு குறிப்பு - நாம் கொடுக்கும் அகல-உயர விகிதம் 1க்குக் குறைவாக இருந்தால், உயரம் இதன்படி குறைக்கப்படும். 1க்கு மேலே இருந்தால்? உயரம் அதிகரிக்கப்படாது. உயரத்தை மாற்றாமல் வைத்துக் கொண்டு 1/given number என்ற (ஒன்றுக்குக் குறைந்த) பின்னத்தால் அகலத்தைப் பெருக்கி, அகலத்தைச் சிறியதாக்கும்.

கொடுக்கும் எண் 1 என்றால் எந்தவித மாற்றமும் செய்யாமல் ஒரு நீள் வட்டத்தை வரையும். அதனால் விரைவில் வரைந்துவிடும். சரியான சில உதாரணங்கள். (இந்த வாக்கியங்களில் எந்தத் தகவல் எதைக் குறிக்கும் என்று கூறு.)

```
CIRCLE (50, 60), 10 ,,,, 3/4 + 0.001
CIRCLE (100, 40), 5, 2, 0, PI, 2.5
CIRCLE (50, 150), 40, 0, 1, 0.5
CIRCLE (10, 20), 5, 1, 2. 5, 1.6
```

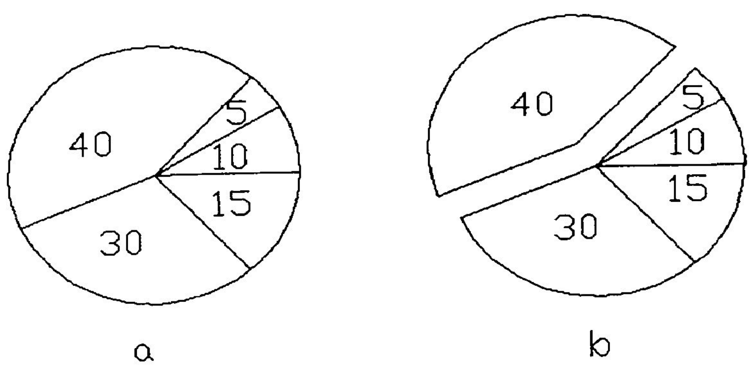
உதாரணங்கள்

1. (100, 100) என்ற புள்ளியில் ஆரம்பித்து, ஒரு வட்டம் சிறிதுசிறிதாகப் பெரிதாகிக் கொண்டே வருவது போல் காண்பி.

இதற்கு வட்டங்களை ஒவ்வொன்றாக (ஆரம் 0 முதல் 50 வரை) வரைந்து, உடனே அழிக்க வேண்டும்.

```
10 SCREEN 1
20 COLOR 0, 1
30 FOR I = 0 TO 50
40      CIRCLE (100, 100), I     : REM DRAW IN COLOR 1
50      CIRCLE (100, 100), I, O : REM ERASE BY COLOR O
60 NEXT I
```

2. வட்டவரைபடம் என்பது, ஒரு வட்டத்தைப் பல பிரிவுகளாக, கொடுத்த சதவிகிதங்களின்படி பிரித்துக் காட்டுவதாகும். 10%, 5%, 40%, 30%, 15% என்று வரிசையாக வட்டத்தைப் பிரிப்போம். படம் 7.6a இதைக் காட்டும்.



படம் 7 - 6 வட்டப் பகுதிகள்

*100 சதவிகிதத்திற்கு 2 π ரேடியன்கள். அதனால் 10 சதவிகிதத்திற்கு 2π * 0.1 ரேடியன்கள்*

```
10 SCREEN 1
20 COLOR 0,0
30 PI = 3.141593
40 CIRCLE (100, 80), 60, 1, 0,-2 * PI * 0.1
50 CIRCLE (100, 80), 60, 1, 2 * PI * 0.1, -2 * PI * 0.15
60 CIRCLE (100, 80), 60, 1, 2 * PI * 0.15, -2 * PI * 0.55
70 CIRCLE (100, 80), 60, 1, 2 * PI * 0.55 , -2 * PI * 0.85
80 CIRCLE (100, 80), 60, 1, 2 * PI * 0.85, -2 * PI
90 END
```

3. மேலே உள்ள வட்ட வரைபடத்தில் ஒரு பகுதியை சற்று வெளிப்புறம் தள்ளி வைத்து, அதைத் தனியாக சிறப்பாகக் காட்டலாம். படம் 7.6 b இல் உள்ளதுபோல் மூன்றாம் பகுதியைத் தனியாகக் காட்டு.

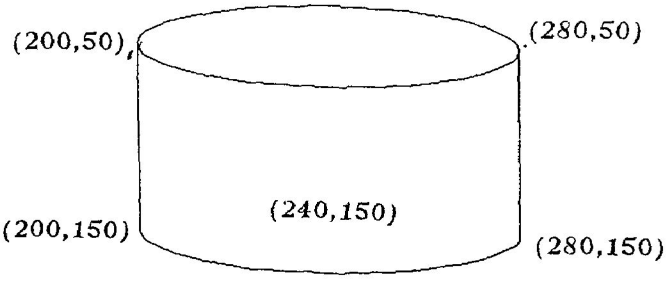
இதற்கு மூன்றாம் பகுதியின் மையப்புள்ளியை, சிறிது தள்ளி வைக்க வேண்டும். இடதுபுறம் 5 புள்ளி, மேற்புறம் 5 புள்ளி தள்ளலாம். இரு முனைகளையும் மையப்புள்ளியுடன் சேர்க்க வேண்டும். இதற்கு 60ஆம் வரியை இப்படி மாற்றலாம்.

```
60 CIRCLE (95, 75), 60, 1, -2 * PI * 0.15, -2 * PI * 0.55
```

70 ஆம் வரியில் ஒரு சிறு மாற்றம் தேவை. அது என்ன? ஏன்?

4. ஒரே மையப்புள்ளி கொண்ட, ஆரங்கள் 5, 10, 15...50 கொண்ட வட்டங்கள் வரை.

```
10 SCREEN 2 : REM HI RESOLUTION. NO COLOR STATEMENT
20 FOR I = 5 TO 50 STEP 5
30       CIRCLE (100, 100), I, 1
40 NEXT I
50 END
```



படம் 7 - 7 உருளை

5. செங்குத்தாக நிற்கும் ஓர் உருளையின் படம் வரை. இந்த உருளையைச் சற்று மேற்புறமாக இருந்து பார்ப்பதுபோல் வரை. இப்போது உருளையின் மேற்பகுதி ஒரு நீள் வட்டமாகத் தெரியும். கீழ்ப்பகுதி படம் 7.7 இல் உள்ளது போல் ஒரு நீள் வட்டத்தில் ஒரு பாதியாகத் தெரியும். மீதி கண்ணுக்குத் தெரியாமல் மறைந்திருக்கும்.

உருளையின் உயரம் 100 புள்ளிகள், ஆரம் 40 புள்ளிகள் என்போம். 40 புள்ளி ஆரமுள்ள வட்டத்தின் உயரத்தைக் குறைக்க 1/4 என்ற அகல-உயர விகிதத்தைப் பயன்படுத்துவோம்.

```
10 SCREEN 1
20 COLOR 0,0
30 LINE (200, 50) - (200, 150)
40 LINE (280, 50 ) - (280, 150)
50 PI = 3.141593
60 CIRCLE (240, 50), 40, 1,,, 0.25 : REM TOP
70 CIRCLE (240, 150), 40, 1, PI, 2 * PI, 0.25 : REM BOTTOM
80 END
```

நினைவில் நிற்க

1.CGA திரையில் தோன்றும் புள்ளிகள் வட்டவடிவமாக இல்லாமல், உயரம் சற்று கூடுதலாக இருக்கும். இந்த அகல-உயர விகிதங்கள் 5/6 மற்றும் 5/12 .

2. வட்டத்தை , அல்லது அதன் பாகத்தை வரையும்போது, உயரத்தை அல்லது அகலத்தை எந்த அளவு குறைக்க வேண்டும் என்பதை நாமே தீர்மானிக்கலாம்.

3. CIRCLE(x, y), radius, color, start angle, end angle, aspect ratio என்னும் வாக்கியம் வட்டம், நீள் வட்டம், அவற்றின் பகுதிகள் என்ற எல்லாவற்றையும் வரையும்.

4. கோணங்கள் ரேடியனில் கொடுக்கப்பட வேண்டும்.

5. வட்டப் பகுதியின் முனைகளிலிருந்து மையத்திற்குக் கோடுகள் வரைய ஒரு கழித்தல் குறி சேர்க்கப்பட வேண்டும். இங்கு -0 என்பதை -2π எனக் குறிப்பிட வேண்டும்.

பயிற்சி வினாக்கள்

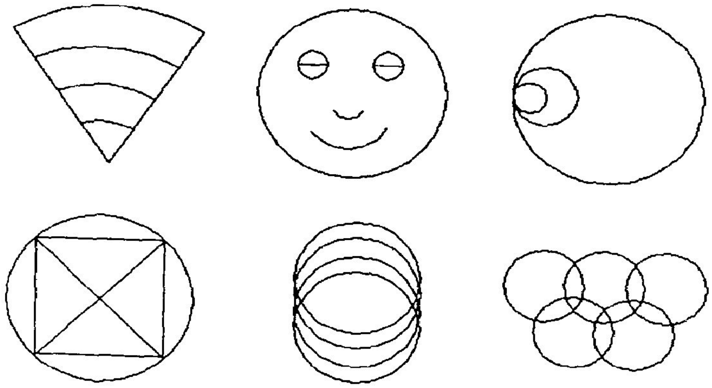
1. CIRCLE என்ற வாக்கியத்தின் இலக்கணத்தைக் கூறு. அதில் கூறப்படும் தகவல்கள் அவசியம் கொடுக்கப்பட வேண்டுமா? கொடுக்கப்படாவிட்டால் என்ன ஆகும்?
2. CIRCLE வாக்கியத்தின் மூலம் வட்டம், நீள்வட்டங்களை வரைவது எப்படி என்று உதாரணங்களுடன் விளக்கு.

3. அகல-உயர விகிதம் என்றால் என்ன? நீள் வட்டங்கள் வரைய அது எப்படிப் பயன்படுகிறது?

4. வட்ட வரைபடம் வரைவது எப்படி என்பதை ஓர் உதாரணத்துடன் விளக்கு.

5. ஒரே மையப்புள்ளி உள்ள வட்டங்கள் இருபதை பல வண்ணங்களில் வரை.

6. கீழ்க்காணும் படங்களை (படம் 7.8)வரைய கட்டளை நிரல்கள் எழுது. முதலில் புள்ளிகளைத் தெரிவு செய்துகொள்.



படம் 7 - 8 வட்டப் படங்கள்

7.8 சலனப்படங்கள்

படங்களை வரையும் விதங்களை இதுவரை பார்த்தோம். அவற்றைத் திரையில் நகரச் செய்வது எப்படி என்பதை இங்கு காண்போம். இதற்கு முதலில் திரையில் வரைந்த படத்தை எடுத்து கணிப்பொறியின் நினைவில் வைத்துக் கொள்ள வேண்டும். பிறகு அந்தப் படத்தை ஓர் இடத்தில் வரைய வேண்டும். உடனே அதை அழிக்க வேண்டும். பிறகு இதற்கு அடுத்த இடத்தில் வரைந்து, பிறகு அழிக்க வேண்டும். இவ்வாறு அடுத்தடுத்த இடங்களில் வரைந்து கொண்டும், அழித்துக் கொண்டும் சென்றால் படம் நகர்வது போல் தெரியும்.

திரையிலிருந்து படத்தை நினைவகத்திற்கு மாற்ற உதவும் கட்டளை GET என்பது. இதன் இலக்கணம்

GET (x1, y1) - (x2, y2), array name

திரையில் உள்ள ஒரு செவ்வகப் பரப்பைத்தான் நினைவில் வைக்க முடியும்.

இந்தச் செவ்வகங்களின் பக்கங்கள் x மற்றும் y அச்சுக்களுக்கு இணையாக இருக்க வேண்டும். அதனால் இந்தச் செவ்வகத்தின் இடது மேல் மூலையும் (x1, y1) வலது கீழ் மூலையையும் (x2, y2) கொடுத்தால் செவ்வகம் வரையறுக்கப்பட்டுவிடும்.

இந்தச் செவ்வகப் பரப்பை நினைவில் வைக்க எவ்வளவு இடம் தேவை? முதலில் செவ்வகத்தில் ஒரு வரிசையில் எத்தனை புள்ளிகள் (x2 - x1 + 1), எத்தனை வரிசைகள் (y2 - y1 + 1) என்ற இரண்டு முழு எண்களைக் குறிக்க 2 + 2 = 4 பைட்டுகள்.

இதற்குப் பிறகு புள்ளிகளின் வண்ணங்கள் வரி வரியாக வைக்கப்படும். உயர் பிரிப்பாற்றலில் 1 புள்ளிக்கு 1 பிட். அதனால் ஒரு வரியில் எட்டு எட்டு புள்ளிகளாகச் சேர்த்து (இடமிருந்து வலமாக) பைட்டுகளாகச் செய்யப்படும். நடுத்தரப் பிரிப்பாற்றலில் ஒரு புள்ளிக்கு 2 பிட்டுகள். அதனால், 4 புள்ளிகள் சேர்ந்து ஒரு பைட்டைக் கொடுக்கும். இங்கு 20 புள்ளிகள் இருந்தால் 20/4 = 5 பைட்டுகள் வேண்டும். 21 புள்ளிகள் இருந்தால்? 6 பைட்டுகள் வேண்டும். கடைசி பைட்டில் சிறிது இடம் வீணாகும். பரவாயில்லை, ஏனென்றால் வரிகளை ஒன்றுடன் ஒன்று இணைக்காமல், தனித்தனியாக வைக்க வேண்டும். தேவையான மொத்த பைட்டுகளின் எண்ணிக்கை

= 4 + (INT(((x2 - x1 + 1) * bits per pixel + 7) / 8)) * (y2 - y1 + 1)

இதில் 7 பிட்டுகளைக் கூட்டுவது, கடைசி பைட்டில் வீணாகக் கூடிய இடத்தையும் சேர்த்துக் கணக்கிடுவதற்காக.

இத்தனை பைட்டுகள் கொள்ளும்படியாக ஓர் அணியின் பெயரை நாம் வரையறை செய்ய வேண்டும். ஓர் உதாரணத்தைப் பார்ப்போம். நடுத்தரப் பிரிப்பாற்றல் திரையில் *(10, 20), (20, 41)* என்னும் புள்ளிகளை எதிரெதிர் மூலைகளாகக் கொண்ட செவ்வகத்தை TOY என்ற வரிசையில் வைக்க வேண்டும்.

தேவையான பைட்டுகள் = 4 + (INT ((11 * 2) + 7) / 8) * 22

= 4 + (INT(29/8)) * 22

= 4 + 3 * 22 = 70

சாதாரண எண்ணுக்கு 4 பைட்டுகள். 18 எண்களுக்கு 18 X 4 = 72 பைட்டுகள். ஆக, குறைந்தது 18 எண்கள் உள்ள வரிசை தேவை. அதற்கு மேல் இருந்தால் தவறில்லை. மீதி இடம் வீணாகும். அவ்வளவுதான்.

```
10 DIM TOY (20)
20 SCREEN 1: REM MEDIUM RES
30 COLOR 0, 0
40 DRAW "BM 10, 20 M 20, 30"
50 REM
60 GET (10, 20) - (20, 41), TOY
```

இப்போது TOY என்ற வரிசையில் ஒரு படம் இருப்பதால் அதை வேறு எதற்கும் பயன்படுத்தக் கூடாது.

நினைவில் நிறுத்திய படத்தை மீண்டும் எப்படித் திரையில் வரைவது? இதற்கு, படம் இருக்கும் வரிசையின் பெயர் தேவை. வரைய வேண்டிய செவ்வக இடத்தின் இடது மேல் மூலைப் புள்ளி தேவை. செவ்வகத்தின் அளவு நினைவில் இருப்பதால் (முதல் இரண்டு எண்கள்) கணிப்பொறி சரியாக இந்தப் படத்தை வரைந்துவிடும். அதனால் , வலது கீழ்ப்புள்ளி கொடுக்கப்பட வேண்டாம்.

அடுத்தது, படத்தை எப்படிப் போட வேண்டும் என்பதைக் கூற வேண்டும். படத்தைப் போடுவதில் 5 வகைகள் உள்ளன. அவை PEST, PRESET, AND, OR மற்றும் XOR எனப்படும். இவற்றில் PSET வகை மிகவும் எளிதானது. படத்தைப் போட வேண்டிய இடத்தில் ஏற்கனவே இருக்கும் படத்தை அழித்து விட்டுப் புதுப் படத்தைப் போட்டுவிடும். அடுத்து வரும் PRESET என்னும் விதத்தில், படமானது புகைப்பட ஃபிலிமில் இருக்கும் நெகடிவ் போல் மாறித் தெரியும். இதற்கு, போட வேண்டிய வண்ணத்தில் உள்ள *0* என்பதை *1* எனவும், *1* என்பதை *0* எனவும், மாற்றிப் போடும். அதனால் *00* என்பது *11* எனவும், *11* என்பது *00* என்றும், *10* என்பது *01* எனவும், *01* என்பது *10* எனவும் மாறும். உயர்பிரிப்பாற்றலில் சுழியும் ஒன்றும் இடம் மாறும்.

மேலே பார்த்த இருவகைகளும், ஏற்கனவே அந்தச் செவ்வகத்தில் இருப்பதைக் கவனிக்காது. ஆனால் அடுத்து வரும் மூன்று வகைகளில், வரும் வண்ணம் ஏற்கனவே இருக்கும் வண்ணத்துடன் கலந்து ஒரு புள்ளியின் புதிய நிறத்தை உருவாக்கும். AND மற்றும் OR செயல்பாடுகளை நாம் ஏற்கனவே பார்த்திருக்கிறோம். XOR மட்டும் புதிது. Exclusive OR என்பதன் சுருக்கம். "a XOR b" என்றால்,"a , b என்ற இரண்டில் ஏதேனும் ஒன்று மட்டும், மற்றது கிடையாது என்பதைக் குறிக்கும். AND, OR, XOR களுக்கான பட்டியல் கீழே உள்ளது.

P	Q	P AND Q	P OR Q	P XOR Q
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

உயர் பிரிப்பாற்றலில் இந்தப் பட்டியலை அப்படியே பயன்படுத்தி ஒரு புள்ளியின் புதிய வண்ணத்தைப் பெறலாம். நடுத்தரப் பிரிப்பாற்றலில் வண்ணத்திற்கு 2 பிட்டுகள் இருக்கும். இந்தப் பட்டியலை முதல் பிட்டிற்குத் தனியாகவும், இரண்டாவது பிட்டிற்குத் தனியாகவும் பயன்படுத்தலாம்.

உதாரணமாக 10 AND 11 *என்பது* 1 AND 1, 0 AND 1 = 10

10 OR 11 *என்பது* 1 OR 1, 0 OR 1 = 11

10 XOR 11 *என்பது* 1 XOR 1, 0 XOR 1 = 01

P AND Q = Q AND P ; P OR Q = Q OR P; P XOR Q = Q XOR P

என்பனவற்றை எளிதாகச் சரிபார்க்கலாம். அதனால் எந்த வண்ணத்தை முதலில் கொடுக்க வேண்டும் என்று விதி எதுவும் இல்லை. இரண்டும் ஒன்றே.

சேமித்து வைத்த படத்தை, திரையில் காட்ட உதவும் கட்டளை PUT. PUT (X, Y) array name , action type என்பது இதன் இலக்கணம். இதில் action type என்பது PSET போன்ற ஐந்தில் ஒன்று. இதை விட்டுவிட்டால் கணிப்பொறி தானாக இதை XORஎன்று வைத்துக் கொள்ளும். சில உதாரணங்கள்-

PUT (100, 10), TOY, PSET

PUT (10, 20), TOY, XOR

PUT (200, 50), TOY

XOR இன் பயன்

XOR வகையில் படம் வரைவது ஒரு பயனுள்ள முறை. அதற்கான காரணம்

(x XOR y) XOR y = x

என்ற சமன்பாடுதான். உதாரணமாக x = 1, y = 0 என்றால்,

(1 XOR 0) XOR 0 = 1 XOR 0 = 1

அதாவது, ஏற்கனவே பின்னணியில் இருக்கும் ஒரு படத்தின் மேல், வேறு ஒரு படத்தை இரண்டு முறைகள் அடுத்தடுத்து XOR வகையில் வைத்தால், பின்னணியில் இருக்கும் படம் சேதமடையாமல் திரும்பக் கிடைத்துவிடும். பின்னணிக் காட்சி சேதமடையாமல், அதன் மேல் ஒரு படம் நகர்ந்து செல்வதைப் போல் காண்பிக்க இம்முறை பெரிதும் உதவும்.

உதாரணமாக, மரங்கள் சில உள்ள பின்னணியில், ஒரு பறவை பறந்து செல்வதைப் போல் காண்பிக்க வேண்டும் எனலாம். இதற்கு முதலில் பறவையை வரைந்து அதை நினைவில் வைத்துக் கொள்ள வேண்டும். பிறகு திரையில் பின்னணிக் காட்சியை வரைய வேண்டும். இப்போது பறவையை ஓர் இடத்தில் இரண்டு முறை XOR விதத்தில் வரைந்தால், அங்கு பறவை தோன்றி மறையும். ஆனால் பறவை மறைந்தவுடன், பின்னணிப் படத்தில் மாற்றம் இருக்காது. அடுத்தடுத்த இடங்களில் பறவை தோன்றி மறைந்தால், பறவை நகர்வதுபோல் தோன்றும். பின்னணிப் படம் மாறாமல் இருக்கும்.

இம்முறையில் ஒரு குறை உண்டு. பின்னணியில் இருக்கும் வண்ணத்துடன் சேர்வதால், நாம் வைக்கும் படம், சில வண்ண மாற்றங்களுடன்தான் தெரியும். இந்த வண்ணமாற்றம், வைக்கும் இடத்தில் ஏற்கனவே உள்ள வண்ணங்களைப் பொறுத்தது.

படங்கள் வேகமாக நகர்ந்து சென்றால், அதன் வேகத்தைக் குறைக்க ஒரு வழி உண்டு. இரண்டு முறைகள் படத்தை வரைவதற்கு நடுவில், நேரத்தைச் செலவிட சில உபயோகமற்ற செயல்களைச் செய்யலாம். உதாரணமாக,

```
FOR JJJ = 1 TO 100
    LLL = 1
NEXT JJJ
```

இப்போது சில கட்டளை நிரல்களை எழுதுவோம்.

1. ஒரு முக்கோணம் திரையின் இடதுபக்கம் தோன்றி வலது பக்கமாக நகர்ந்து செல்வதுபோல் காண்பிக்கலாம்.

1. திரையைத் தயார் செய்.

2. ஒரு முக்கோணம் வரை.

3. அதை நினைவில் வை.

4. முக்கோணத்தை அதே இடத்தில் XOR வகையில் வை (அழி).

5. 0 முதல் 200 வரை செய்

புதிய இடத்தில் XORவகையில் முக்கோணத்தை வை.

அதே இடத்தில் XOR வகையில் திரும்ப வை (அழி).

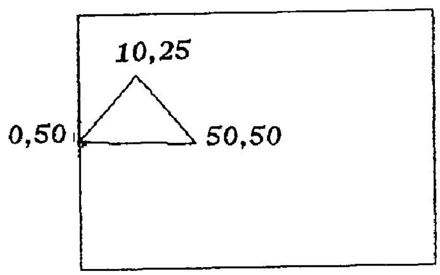
6. முடிவு.

ஆரம்பத்தில் முக்கோணம் இப்படி (படம் 7.9) இருக்கட்டும்.

இந்தப் படம் இருக்கும் செவ்வகம் (0, 25) - (50, 50). நடுத்தரப் பிரிப்பாற்றல் திரையில் இதற்குத் தேவையான இடம்

4 + (INT ((51 * 2 + 7) /8)) * 26 = 4 + (INT (109/8)) * 26

= 4 + 13 * 26 = 342 பைட்டுகள்.



படம் 7 - 9 நகரும் முக்கோணம்

```
10  DIM TRI (100)
20  SCREEN 1: REM MED RESOLUTION
30  COLOR 0, 0
40  REM
50  DRAW "BM 10, 25 M 50, 50 M 0, 50 M 10, 25"
60  GET (0, 25 ) - (50, 50), TRI
70  PUT (0, 25), TRI, XOR : REM ERASE
80  REM
100 FOR I = 0 TO 200
110     PUT (I, 25) TRI, XOR
120     PUT (I, 25), TRI, XOR : REM ERASE
130 NEXT I
140 END
```

2. ஒரு சிறு சதுரத்தை வரை. அதை முறையில்லாமல் 10 இடங்களில் வை.

```
10  SCREEN 1
20  COLOR 0,0
30  DRAW "BM 0 , 0 M 0, 20 M 20, 20 M 20, 0 M 0, 0"
40  GET (0 ,0) - (20, 20), SQUR
50  FOR I = 1 TO 10
60      PUT (INT (RND * 300), INT (RND * 180)), SQU, PSET
70  NEXT I
80  END
```

இங்கு இரண்டு முக்கிய வரிகளை விட்டுவிட்டோம். அவை என்ன?

RANDOMIZE *மற்றும்* DIM *வாக்கியங்கள்*

5 DIM SQUR (200)*என்று கொடுத்தால் போதுமா என்று பார்.*

3. திரையின் நடுவில் ஒரு சிறு + குறியீடு வரை. அதை நினைவில் வை. திரையில் எந்த இடத்தில் வைக்க வேண்டும் என்று கேட்டு அங்கு PSET*வகையில் வரை.*

```
10 SCREEN : COLOR 3, 0
20 DIM PLUS (50)
30 X = 160 : Y = 100
40 DRAW "BM X - 5, Y M X + 5, Y B M X, Y - 5 M X, Y + 5"
50 GET (X - 5, Y - 5) - (X + 5, Y + 5), PLUS
60 PUT (X - 5, Y - 5), PLUS, XOR : REM ERASE
70 INPUT "TELL WHERE TO PUT? GIVE X AND Y" ; X,Y
80 PUT (X,Y), PLUS, PSET
90 END
```

நினைவில் நிற்க

1. படத்தை நினைவில் நிறுத்த GET என்னும் வாக்கியம் பயன்படும். இதில் செவ்வகத்தின் இடது மேல்மூலை, வலது கீழ் மூலை, வரிசையின் பெயர் கொடுக்கப்பட வேண்டும்.

2. படத்தைத் திரையில் காட்ட PUT வாக்கியம் பயன்படும். இதில் செவ்வகத்தின் இடது மேல் மூலை, வரிசையின் பெயர் மற்றும் வைக்கும் விதம் கொடுக்க வேண்டும்.

3. PSET, PRESET என்பவை இருப்பதை அழித்துவிடும்.

4. AND, OR, XOR என்பவை ஏற்கனவே இருக்கும் படத்துடன் கலந்து வரும்.

5. ஒரு படத்தின் மேல் இன்னொரு படத்தை இருமுறை XOR வகையில் வைத்தால், பின்னணிப்படம் சிதையாமல் திரும்பக் கிடைத்துவிடும்.

பயிற்சி வினாக்கள்

1. GETமற்றும் PUTவாக்கியங்களின் பொது வடிவங்களை விவரி.

2. AND, OR , XOR இவற்றின் உண்மைப் பட்டியலை எழுது.

3. ஒரு படத்தையே PUT வாக்கியத்தைப் பயன்படுத்தி 5 விதங்களில் வரையலாம். எவ்வாறு?

4. கீழே உள்ள சலனப் படங்களுக்கான கட்டளை நிரல்களை எழுது. கொடுக்கப்படாத செய்திகளை நீயே முடிவு செய்து கொள்.

அ. ஒரு செவ்வகத்தையும் இரண்டு வட்டங்களையும் பயன்படுத்தி ஒரு பேருந்தின் படம் வரை. அது இடமிருந்து வலமாக நகரட்டும்.

ஆ. ஒரு பந்து மேலிருந்து கீழே விழுந்து பிறகு எழும்புவதுபோல் காட்டு.

இ. ஒரு வட்டம் சிறிது சிறிதாக அளவில் குறைந்து ஒரு புள்ளியாக மாறிவிட்டு, அதிலிருந்து திரும்பவும் பெரிய வட்டமாக சிறிது சிறிதாக மாறட்டும்.

ஈ. ஒரு சதுரத்தையும், அதை நான்கு பக்கங்களிலும் தொடும் ஒரு வட்டத்தையும் வரை. ஒன்று மாற்றி ஒன்றாகக் காண்பி.

உ. ஒரு மகிழ்ச்சியான முகம் வரை. சிறிதுநேரம் கழித்து அதை சோக முகமாக மாற்று. இவ்வாறு மாற்றி மாற்றி 100 முறை செய்.

5. கீழே உள்ள வாக்கியங்களில் தவறுகள் இருந்தால் கண்டுபிடி.

அ. GET (2, 3), PIC

ஆ. GET (100, 100), (50, 50), WEL

இ. GET (50, 70) - (70, 0) W, PSET

ஈ. GET (60, - 10) - (100, 100), TOY, XOR

உ. PUT MAX, 0, 100

ஊ. PUT O, 100, MAX, XOR

எ. PUT MAX, XOR

ஏ. PUT MAX (O, 5) - (20, 30), PRESET

ஐ. PUT (0, 0), MAX, AND

ஒ. PUT (5, 5) MAX OR

6. ஒரு குறிப்பிட்ட செவ்வகத்தைச் சேமிக்கத் தேவையான நினைவகத்தின் அளவை எவ்வாறு கண்டுபிடிப்பது? கணக்கீட்டை ஓர் உதாரணத்தின் மூலம் விவரி. அதற்கான DIM வாக்கியத்தையும் எழுது.

7.9 எழுத்துத் திரையும் ஓசையும்

சலனப்படங்கள் திரையில் தோன்றும் போது அவற்றுக்கேற்ற பின்னணி இசையும் இருந்தால் சிறப்பாக இருக்குமல்லவா? ஓசை பிறப்பிக்கும் ஒரு கட்டளையை மட்டும் இங்கு பார்ப்போம்.

SOUND frequency, duration

என்ற வாக்கியத்தில் frequency என்பது 37 முதல் 32,767 வரை இருக்கலாம். duration என்பது ஓசை வரும் நேரம். இது 0 முதல் 65,535 வரை இருக்கலாம். இதில் 1 என்பதற்கு மதிப்பு 1/18.2 விநாடிகள். அதாவது 18.2 என்பது 1 விநாடியைக் குறிக்கும். உதாரணமாக,

SOUND 400, 364 என்பது 400 அதிர்வுகளில் இருபது விநாடிகள் ஒலி எழுப்பும். DRAW வாக்கியம் போல், பாடல்களை இசைக்க PLAY என்று ஒரு வாக்கியம் இருக்கிறது. ஆர்வமுள்ளவர்கள் அதுபற்றி அறிந்து கொள்ளலாம்.

எழுத்துத்திரை

படத்திரையில் பல வண்ணங்களில் படங்கள் வரைவதைப் பார்த்தோம். எழுத்துகளையும் பல வண்ணங்களில் காண்பிப்பது பயனுள்ளது. இதன் மூலம்முக்கிய விவரங்களைத் தனிவண்ணத்தில் மற்றவற்றிலிருந்து வேறுபடுத்திக் காட்டலாம். தகவல்களைப் பல பகுதிகளாகப் பிரித்துக் காட்டலாம். பார்ப்பதற்குத் திரை அழகாக இருக்கும். உபயோகிப்பவரைக் கவரும் விதத்திலும் இருக்கும். இதற்கான சில வாக்கியங்களை இங்கே பார்ப்போம்.

SCREEN 0, 1

என்பதில் வரும் 0 என்பது எழுத்துத்திரையையும், அடுத்துவரும் 1 என்பது இந்தத் திரையில் எழுத்துகள் வண்ணம் பெறலாம் என்பதையும் குறிக்கும். அடுத்து WIDTH 80 என்பது வரிக்கு 80 எழுத்துக்கள் உள்ள திரையையும் WIDTH 40 என்பது வரிக்கு 40 எழுத்துக்கள் உள்ள திரையையும் உருவாக்கும். இவ்வாறு தெரிவு செய்யும் போது ஒரு விதத்திலிருந்து இன்னொரு விதத்திற்குச் சென்றால் திரை அழிக்கப்பட்டு புதுத்திரை தோன்றும்.

படத்திரையில் COLOR வாக்கியம், முழுத்திரையின் பின்னணி வண்ணத்தையும், வண்ணத்தட்டையும் தெரிவு செய்யும். எழுத்துத் திரையில் இருக்கும்போது இது ஒவ்வொரு எழுத்துக்கும் **தனியாக** உள்ள முன்னணி மற்றும் பின்னணி வண்ணங்களைத் தெரிவு செய்யும்.

COLOR foreground, background

என்பது இதன் இலக்கணம். இந்த இரண்டு தகவல்களில் எதையும் விட்டுவிடலாம். அப்போது விடுபட்ட தகவலுக்கு ஏற்கனவே உள்ள மதிப்பு அப்படியே மாறாமல் நீடிக்கும்.

COLOR 3, 0 COLOR 6 COLOR , 2

என்பவை சரியான வாக்கியங்கள். ஒவ்வொரு எழுத்திற்கும் பின்னணி என்பது 8 X 8 அல்லது 8 X 16 புள்ளி செவ்வகத்தைக் குறிக்கும். எழுத்துகள் எழுதப்படும் வண்ணம் முன்னணி வண்ணம் எனப்படும். பின்னணி வண்ணம் 0 முதல் 7 வரையிலும், முன்னணி வண்ணம் 0 முதல் 15 வரையிலும் இருக்கலாம்.

ஒரு COLOR வாக்கியத்தில் கொடுத்த மதிப்புகள், அடுத்த COLOR வாக்கியம் வரும் வரை தொடரும். முன்னணி வண்ணத்துடன், பதினாறினைச் சேர்த்து எழுதினால் (இப்போது இந்த எண்கள் 16 முதல் 31 வரை இருக்கும்) எழுத்துக்கள் தோன்றித் தோன்றி மறைந்து மினுக்கும்.

ஒரு உதாரணம்

```
10 SCREEN 0, 1: REM TEXT SCREEN WITH COLORS
20 WIDTH 40
30 COLOR 2, 0 : REM BLACK BACKGROUND. GREEN FOREGD
40 PRINT "WELCOME"
50 COLOR 1, 7 : REM BLUE ON WHITE BACKGROUND
60 PRINT "TO"
70 COLOR 5 + 16, 7 : REM MAGENTA BLINKS ON WHITE
80 PRINT "ANNA UNIVERSITY"
90 REM "ANNA UNIV" BLINKS
100 END
```

எழுத்துத் திரையை அழிக்க CLS என்பதைப் பயன்படுத்தலாம். திரை சுத்தமாகி இடம் சுட்டி முதல் வரியில் முதல் இடத்திற்குப் போகும். படத்திரைக்கும் CLS பயன்படுத்தலாம். திரை சுத்தமாகும். திரையின் மையப்புள்ளி தற்போதைய இடமாக மாறும்.

நினைவில் நிற்க

1. ஓசையை, குறிப்பிட்ட அதிர்வெண்களில் குறிப்பிட்ட நேரத்திற்கு எழுப்பலாம்.

2. எழுத்துத் திரையில் வண்ணம் கொண்டுவரலாம். ஒவ்வொரு எழுத்துக்கும், தனித்தனியாகப் பின்னணி, முன்னணி வண்ணங்களைத் தேர்ந்தெடுக்கலாம்.

3. எழுத்துக்களை மின்னச் செய்யலாம்.

பயிற்சி வினாக்கள்

1. உங்கள் கல்லூரி/நிறுவனத்தின் பெயரை, பல வரிகளில், பல வண்ணங்களில் எழுது, சிலவற்றை மின்னும்படி செய்.
2. எழுத்துத் திரையில் பின்னணி வண்ணம் சிவப்பாகவும், முன்னணி வண்ணம் பச்சையாகவும் இருக்கும்போது இடைவெளிச் சாவியை பல முறை தட்டினால் என்ன வரும்?
3. எழுத்துத்திரையில் COLOR வாக்கியம் பயன்படும் விதத்தை விவரி.
4. CLS என்பதன் செயல்பாட்டை விவரி.

கலைச்சொல் பட்டியல் 1

அகல உயர விகிதம்	aspect ratio
அடுக்குக் குறியீடு	exponential notation
அண்ட் கதவு	AND gate
அமைப்பின் கட்டளை	system command
அழியா வகை	non volatile
அழியும் நினைவகம்	volatile memory
ஆர் கதவு	OR gate
ஆஸ்கி	ASCII
இடம் சார்ந்த குறியீடு	positional notation
இடைஇடம்	buffer
இயக்க அமைப்பு	operating system
இரட்டைப் படைச் சரிபார்க்கும் முறை	even parity check
இருநிலை	binary
இருமடங்கு அடர்த்தி	double density
இருமடங்கு துல்லிய மாறி	double precision variable
இரு வழிப் பிரிதல்	two way branching
இலக்கணப் பிழை	syntax error
இலக்க வகை	digital
இலக்கு நிரல்	object code
இன்வெர்ட்டர்	invertor
ஈருறுப்புச் செயல்பாடு	binary operation
உடனடியற்ற முறை	offline
உடனடி முறை	online
உயர்நிலை கணிப்பொறி மொழி	high level language
உயர் அடர்த்தி	high density
உயர் கணிப்பொறி	Super Computer
உருமாற்றம்	morphing
உலா வருதல்	walk through
உள்நினைவகம்	internal memory
உள்ளீட்டகம்	input unit
உறவுச் செயல்பாடு	relational operator

உறவு நிலைத்தகவல் அமைப்பு	relational data base
எண்ணிலை	octal
எண்ணும்மாறி	index variable
எண்மானக் கோவை	numerical expression
எல்லைக் கோடு	delimiter
ஒப்புமை வகை	analog
ஒருங்கிணைந்த சுற்று	integrated circuit
ஒருமடங்கு துல்லிய மாறி	single precision variable
ஒருவரிச் சார்பு	one line funcion
ஒலிப் பேழை	audio cassette
ஒளிப் பேழை	video cassette
ஒளிப்பேனா	light pen
ஒளியியல் படிப்பான்	optical character reader
ஒற்றைப் படைச்சரி பார்க்கும் முறை	odd parity check
ஓர்உறுப்புச் செயல்பாடு	unary operation
கட்டளை நிரலை நினைவில் வைக்கும் முறை	stored program concept
கட்டளை நிரல்	program
கட்டுப்பாட்டகம்	control unit
கட்டுப்பாட்டு அமைப்புகள்	control structures
கட்டுப்பாட்டு வாக்கியம்	control statement
கணித தர்க்கச் செயலகம்	arithmetic logic unit
கணிப்பான்	calculator
கலப்பின வகை	hybrid
காந்த எழுத்துப் படிப்பான்	magnetic ink character reader
காந்த வளையம்	magnetic core
கீழ்க்குறியீடு	suffix

குறிப்பு வரி	remark
குறுக்கீடு	interrupt
குறை கடத்தி	semi conductor
கூட்டுவான்	assembler
கூட்டு மொழி	assembly language
கெபாசிடர்	capacitor
கோப்பு	file
சமதள வரைவான்	flat bed plotter
சரக் கோவை	string expression
சரம்	string
சார்பு சாவி	function key
சாவிப் பலகை	keyboard
சில்லு	chip
சிறப்புக் குறியீடு	special symbol
சிறப்புச் சாவி	special key
சிறுபரப்புப் பின்னலமைப்பு	Local Area Network
சிறு கணிப்பொறி	mini computer
சுட்டி	cursor
சுண்டெலி	mouse
சுற்று	circuit
செயல்முறை	algorithm
செயற்கை அறிவு	artificial intelligence
சொந்தக் கணிப்பொறி	personal computer
சொல் தொகுப்பான்	text editor
சொல் நீளம்	word length
டிராக்கர் பந்து	tracker ball
டிரான்ஸிஸ்டர்	transistor
தடுப்பான்	resistor
தர்க்கச் செயல்பாடு	logical operation
தற்போதைய இடம்	current position
தனிமதிப்பு	absolute value
தாய்ப்பலகை	mother board
திசைச் சாவி	arrow key, direction key
திரைச் சாதனம்	monitor

தொகுப்பான்	compiler
தொகுப்பு முறை	batch processing
தொடர் வரிசைப் படுத்தல்	concatenation
தொடு திரை	touch screen
நகரும் சட்டம்	slide rule
நிகழ் காலச் செயல்பாட்டு முறை	real time processing
நிச்சயமற்று செய்தல்	indefinite iteration
நிச்சயித்து செய்தல்	definite iteration
நிபந்தனை	condition
நிபந்தனைக் கோவை	conidtional expression
நிபந்தனையின்றிப் பிரிதல்	unconditional branching
நிரப்பு முறை	complement notation
நிரலர்	programmer
நிலைப் பொருள்	firmware
நினைவகம்	memory
நினைவூட்டு	prompt
நீள்வட்டம்	ellipse
நுண்	micro
நூலகச் சார்பு	library function
நெகிழ் தகடு	floppy disc
நேப்பியர் குச்சிகள்	napier bones
நேரடி அணுகுமுறைக் கோப்பு	random access files
நேரடி அணுகு முறை	random access
நேரப் பகிர்வு முறை	time sharing
பகுதி	sector
பட்டை வரி	bar code
பட்டை வரைபடம்	bar chart
படி	hard copy
படிக்க மட்டும் நினைவகம்	read only memory
பதிவிடம்	register

பதிவேடு	record
பதினாறு நிலை	hexadecimal
பல் ஊடகம்	multimedia
பலபணி முறை	multi tasking
பல முறை செய்தல்	looping
பல வழிப் பிரிதல்	multiway branching
பலவரிச் சார்பு	multiline function
பஸ்	bus
பிட்	bit
பிரதான கணிப்பொறி	mainframe computer
பிரிதல்	branching
பிழைச் செய்தி	error message
பிழை நீக்கி	debugger
பின்னணி வண்ணம்	background color
பின்னலமைப்பு	network
பிரிப்பாற்றல்	resolution
புள்ளி வரிசை அச்சுப் பொறி	dot matrix printer
பூலியன் கணிதம்	boolean algebra
பூலியன் கோவை	boolean expression
பெரும் பரப்புப் பின்னலமைப்பு	Wide Area Network
பைட்	byte
பொறி மொழி	Machine Language
போலிக் குறியீடு	pseudo code
போலி முறையில்லா எண்	pseudo random number
மணிச்சட்டம்	abacus
மதிப்பளிக்கும் வாக்கியம்	assignment statement
மதிப்பு	value
மாயத் தோற்றம்	virtual reality
மாற்றுசாவி	shift key
மாறி	variable
மாறிலி	constant
மிதக்கும் புள்ளி முறை	floating point notation
மின்னணுக் கணிப்பொறி	electronic computer
மின்னணுப் பலகை	electronic tablet
முகவரி	address
முழுஎண் மாறி	integer variable
முறையில்லா எண்	random number
முன்னுரிமை	hierarchy
முனையம்	terminal
மூல எண்	seed
மூல நிரல்	source code
மென்பொருள்	software
மேஜைப் பதிப்பு	desk top publishing
மையச் செயலகம்	central processing unit
மை பீச்சு அச்சுப் பொறி	ink jet printer
மோடம்	modem
வட்ட வரைபடம்	pie chart
வண்ணத் தட்டு	palette
வரிசை முறை	sequence
வரிசையாக அணுகும் முறை	sequential access
வரிசை அணுகுமுறைக் கோப்பு	sequential access file
வரி அச்சுப் பொறி	line printer
வரி எண்	line number
வரி மொழி மாற்றி	interpreter
வரைவான்	plotter
வழிப்படம்	flow chart
வளையம்	loop
விரல் சக்கரம்	thump wheel
விரிவு	extension
வெண் பழுப்பு	softwhite
வெளியீட்டகம்	output unit
வெளி நினைவகம்	external memory
வெற்றிடக் குழாய்	vacuum tube
ஜாய் ஸ்டிக்	joy stick

கலைச்சொல் பட்டியல் 2

Abacus	மணிச்சட்டம்
Absolute value	தனி மதிப்பு
Address	முகவரி
Algorithm	செயல் முறை
Analog	ஒப்புமை வகை
AND gate	அண்ட் கதவு
Arithmetic Logic Unit	கணித தர்க்கச் செயலகம்
Arrow key, Direction key	திசைச் சாவி
Artificial intelligence	செயற்கை அறிவு
ASCII	ஆஸ்கி
Aspect ratio	அகல உயர விகிதம்
Assembler	கூட்டுவான்
Assembly Language	கூட்டு மொழி
Assaignment statement	மதிப்பளிக்கும் வாக்கியம்
Audio cassette	ஒலிப் பேழை
Background Color	பின்னணி வண்ணம்
Bar Chart	பட்டை வரைபடம்
Bar Code	பட்டை வரி
Batch processing	தொகுப்பு முறை
Binary	இருநிலை
Binary operation	ஈருறுப்புச் செயல்பாடு
Bit	பிட்
Boolean algebra	பூலியன் கணிதம்
Boolean expression	பூலியன் கோவை
Branching	பிரிதல்
Buffer	இடை இடம்
Bus	பஸ்
Byte	பைட்
Calculator	கணிப்பான்
Capacitor	கெபாசிட்டர்
Central Processing Unit	மையச் செயலகம்
Chip	சில்லு
Circuit	சுற்று
Compiler	தொகுப்பான்
Complement notatin	நிரப்பு முறை
Concatenation	தொடர் வரிசைப்படுத்தல்
Condition	நிபந்தனை
Conditional expression	நிபந்தனைக் கோவை
Constant	மாறிலி
Control statement	கட்டுப்பாட்டு வாக்கியம்
Control structures	கட்டுப்பாட்டு அமைப்புகள்
Control Unit	கட்டுப்பாட்டகம்
Current position	தற்போதைய இடம்
Cursor	சுட்டி
Debugger	பிழை நீக்கி
Definite iteration	நிச்சயித்து செய்தல்
Delimiter	எல்லைக் குறியீடு
Desk Top Publishing	மேஜைப் பதிப்பு
Digital	இலக்க வகை
Dot matrix printer	புள்ளி வரிசை அச்சுப் பொறி
Double density	இருமடங்கு அடர்த்தி
Double precision variable	இருமடங்கு துல்லிய மாறி
Electronic computer	மின்னணுக் கணிப்பொறி
Electronic tablet	மின்னணுப் பலகை
Ellipse	நீள்வட்டம்
Error message	பிழைச் செய்தி
Even parity check	இரட்டைப்படைச் சரி பார்க்கும் முறை
Exponential notation	அடுக்குக் குறியீடு
Extension	விரிவு
External memory	வெளி நினைவகம்
File	கோப்பு
Firmware	நிலைப் பொருள்
Flat bed plotter	சமதள வரைவான்

Floating point notation	மிதக்கும் புள்ளி முறை
Floppy disc	நெகிழ் தகடு
Flow chart	வழிப்படம்
Function key	சார்பு சாவி
Hard copy	படி
Hierarchy	முன்னுரிமை
Hexadecimal	பதினாறு நிலை
High density	உயர் அடர்த்தி
High level language	உயர் நிலை மொழி
Hybrid	கலப்பின வகை
Indefinite iteration	நிச்சயமற்று செய்தல்
Index variable	எண்ணும் மாறி
Ink jet printer	மை பீச்சு அச்சுப் பொறி
Input unit	உள்ளீட்டகம்
Integer variable	முழுஎண் மாறி
Integrated circuit	ஒருங்கிணைந்த சுற்று
Internal memory	உள்நினைவகம்
Interpreter	வரி மொழி மாற்றி
Interrupt	குறுக்கீடு
Invertor	இன்வெர்ட்டர்
Joy stick	ஜாய் ஸ்டிக்
Keyboard	சாவிப் பலகை
Library function	நூலகச் சார்பு
Light pen	ஒளிப் பேனா
Line number	வரிஎண்
Line printer	வரி அச்சுப் பொறி
Local Area Network	சிறுபரப்புப் பின்னலமைப்பு
Logical operation	தர்க்கச் செயல்பாடு
Loop	வளையம்
Looping	பலமுறை செய்தல்
Machine Language	பொறி மொழி
Magnetic core	காந்த வளையம்
Magnetic Ink Character Reader	காந்த எழுத்துப் படிப்பான்

Mainframe Computer	பிரதான கணிப்பொறி
Memory	நினைவகம்
Micro	நுண்
Mini Computer	சிறு கணிப்பொறி
Modem	மோடம்
Monitor	திரைச்காதனம்
Morphing	உருமாற்றம்
Mother board	தாய்ப்பலகை
Mouse	சுண்டெலி
Multiline function	பல வரிச் சார்பு
Multimedia	பல் ஊடகம்
Multiway branching	பலவழிப் பிரிதல்
Multi tasking	பலபணி முறை
Napier Bones	நேப்பியர் குச்சிகள்
Network	பின்னலமைப்பு
Non volatile	அழியா வகை
Numerical expression	எண்மானக் கோவை
Object code	இலக்கு நிரல்
Octal	எண்ணிலை
Odd parity check	ஒற்றைப் படைச் சரிபார்க்கும் முறை
Offline	உடனடியற்ற முறை
One line function	ஒருவரிச் சார்பு
Online	உடனடி முறை
Operating System	இயக்க அமைப்பு
Optical character reader	ஒளியியல் படிப்பான்
OR gate	ஆர் கதவு
Output Unit	வெளியீட்டகம்
Palette	வண்ணத் தட்டு
Personal Computer	சொந்தக் கணிப்பொறி
Pie chart	வட்ட வரைபடம்
Plotter	வரைவான்
Positional notation	இடம் சார்ந்த குறியீடு
Program	கட்டளை நிரல்
Programmer	நிரலர்

Prompt	நினைவூட்டு
Pseudo random number	போலிமுறையில்லா எண்
Pseudo code	போலிக் குறியீடு
Random access	நேரடி அணுகுமுறை
Random access file	நேரடி அணுகு முறைக் கோப்பு
Random number	முறையில்லா எண்
Read Only Memory	படிக்க மட்டும் நினைவகம்
Real time processing	நிகழ்காலச் செயல்பாட்டு
Record	முறை பதிவேடு
Register	பதிவிடம்
Relational data base	உறவுநிலைத் தகவல் அமைப்பு
Relational operator	உறவுச் செயல்பாடு
Remark	குறிப்பு வரி
Resistor	தடுப்பான்
Resolution	பிரிப்பாற்றல்
Sector	பகுதி
Seed	மூல எண்
Semi conductor	குறை கடத்தி
Sequence	வரிசை முறை
Sequential access	வரிசையாக அணுகும் முறை
Sequential access file	வரிசை அணுகுமுறைக் கோப்பு
Shift key	மாற்று சாவி
Single precision variable	ஒருமடங்கு துல்லிய மாறி
Slide rule	நகரும் சட்டம்
Software	மென்பொருள்
Softwhite	வெண் பழுப்பு
Source code	மூல நிரல்
Special key	சிறப்புச் சாவி
Special symbol	சிறப்புக் குறியீடு
Stored program concept	கட்டளை நிரலை நினைவில் வைக்கும் முறை
String	சரம்
String expresion	சரக் கோவை
Suffix	கீழ்க் குறியீடு
Super Computer	உயர் கணிப்பொறி
Syntax error	இலக்கணப் பிழை
System command	அமைப்பின் கட்டளை
Terminal	முனையம்
Text editor	சொல் தொகுப்பான்
Thumb wheel	விரல் சக்கரம்
Time sharing	நேரப் பகிர்வு முறை
Touch screen	தொடு திரை
Tracker ball	டிராக்கர் பந்து
Transistor	டிரான்ஸிஸ்டர்
Two way branching	இரு வழிப் பிரிதல்
Unary operation	ஓர்உறுப்புச் செயல்பாடு
Unconditional branching	நிபந்தனையின்றிப் பிரிதல்
Vacuum tube	வெற்றிடக் குழாய்
Value	மதிப்பு
Variable	மாறி
Video cassette	ஒளிப் பேழை
Virutal reality	மாயத் தோற்றம்
Volatile memory	அழியும் நினைவகம்
Walk through	உலா வருதல்
Wide Area Network	பெரும்பரப்புப் பின்னலமைப்பு
Word length	சொல் நீளம்

ரூ. 70.00

பாவை பிரிண்டர்ஸ் (பி) லிமிடெட்
142, ஜானி ஜான் கான் ரோடு,
ராயப்பேட்டை,
சென்னை - 600 0014.